

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A209 231



THESIS

DTIC
ELECTE
JUN 22 1989
S
E
D

AN APPLICATION OF A KALMAN FILTER FIXED
INTERVAL
SMOOTHING ALGORITHM TO UNDERWATER
TARGET TRACKING

by

Richard B. Nicklas

March 1989

Thesis Advisor

Harold A. Titus

Approved for public release; distribution is unlimited.

89 6 20 032

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified			1b Restrictive Markings		
2a Security Classification Authority			3 Distribution Availability of Report		
2b Declassification Downgrading Schedule			Approved for public release; distribution is unlimited.		
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)		
6a Name of Performing Organization Naval Postgraduate School		6b Office Symbol (if applicable) 62	7a Name of Monitoring Organization Naval Postgraduate School		
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000			7b Address (city, state, and ZIP code) Monterey, CA 93943-5000		
8a Name of Funding Sponsoring Organization		8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number		
8c Address (city, state, and ZIP code)			10 Source of Funding Numbers		
			Program Element No	Project No	Task No
			Work Unit Accession No		
11 Title (include security classification) AN APPLICATION OF A KALMAN FILTER FIXED INTERVAL SMOOTHING ALGORITHM TO UNDERWATER TARGET TRACKING					
12 Personal Author(s) Richard B. Nicklas					
13a Type of Report Master's Thesis		13b Time Covered From To		14 Date of Report (year, month, day) March 1989	
				15 Page Count 66	
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)		
Field	Group	Subgroup	Kalman Filter, smoothing algorithm, torpedo tracking.		
19 Abstract (continue on reverse if necessary and identify by block number)					
<p>A Fortran program was developed to implement a Kalman Filter and Fixed Interval Smoothing Algorithm to optimally smooth data tracks generated by the short base-line tracking ranges at the Naval Torpedo Station, Keyport, Washington. The program is designed to run on a personal computer and requires as input a data file consisting of X, Y, and Z position coordinates in sequential order. Data files containing the filtered and smoothed estimates are generated by the program. This algorithm uses a second order linear model to predict a typical target's dynamics. The program listings are included as appendices.</p> <p>Several runs of the program were performed using actual range data as inputs. Results indicate that the program effectively reduces random noise, thus providing very smooth target tracks which closely follow the raw data. Tracks containing data generated in an overlap region where one array hands off the target to the next array are highlighted. The effects of varying the magnitude of the excitation matrix $Q(k)$ are also explored.</p> <p>This program is seen as a valuable post-data analysis tool for the current tracking range data. In addition, it can easily be modified to provide improved real time, on line tracking using the Kalman Filter portion of the algorithm alone.</p>					
20 Distribution Availability of Abstract			21 Abstract Security Classification		
<input checked="" type="checkbox"/> unclassified unlimited <input type="checkbox"/> same as report <input type="checkbox"/> DTIC users			Unclassified		
22a Name of Responsible Individual Harold A. Titus			22b Telephone (include Area code) (408) 646-2560		22c Office Symbol 62TS

DD FORM 1473,84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

An Application of a Kalman Filter Fixed Interval
Smoothing Algorithm to Underwater Target Tracking

by

Richard B. Nicklas
Lieutenant, United States Navy
B.S.E.E., United States Naval Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

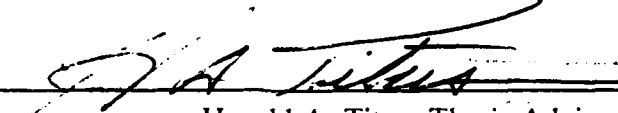
NAVAL POSTGRADUATE SCHOOL
March 1989

Author:



Richard B. Nicklas

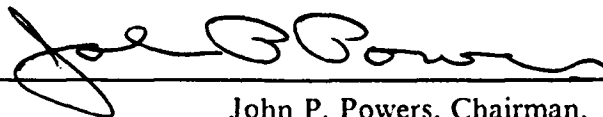
Approved by:



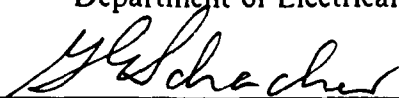
Harold A. Titus, Thesis Advisor



Jeff Burl, Second Reader



John P. Powers, Chairman,
Department of Electrical Engineering



Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

A Fortran program was developed to implement a Kalman Filter and Fixed Interval Smoothing Algorithm to optimally smooth data tracks generated by the short base-line tracking ranges at the Naval Torpedo Station, Keyport, Washington. The program is designed to run on a personal computer and requires as input a data file consisting of X, Y, and Z position coordinates in sequential order. Data files containing the filtered and smoothed estimates are generated by the program. This algorithm uses a second order linear model to predict a typical target's dynamics. The program listings are included as appendices.

Several runs of the program were performed using actual range data as inputs. Results indicate that the program effectively reduces random noise, thus providing very smooth target tracks which closely follow the raw data. Tracks containing data generated in an overlap region where one array hands off the target to the next array are highlighted. The effects of varying the magnitude of the excitation matrix $Q(k)$ are also explored.

This program is seen as a valuable post-data analysis tool for the current tracking range data. In addition, it can easily be modified to provide improved real time, on line tracking using the Kalman Filter portion of the algorithm alone.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



TABLE OF CONTENTS

I. INTRODUCTION	1
II. KALMAN FILTER	8
III. FIXED INTERVAL SMOOTHING ALGORITHM.	18
IV. SOLVING THE ARRAY HANDOFF PROBLEM.	26
V. CONCLUSION	43
APPENDIX A. KALMAN FILTER FIXED INTERVAL SMOOTHING AL- GORITHM	45
A. MAIN PROGRAM	45
B. SUBROUTINES	45
APPENDIX B. INPUT DATA FILE FORMATTING PROGRAM	55
LIST OF REFERENCES	57
INITIAL DISTRIBUTION LIST	59

LIST OF FIGURES

Figure 1.	Short Base-line Array Range Configuration - Fig 2 From Ref 1.	2
Figure 2.	Short Base-line Array Configuration.	3
Figure 3.	Typical Track Generated By A Single Array (X vs Y).	4
Figure 4.	Typical Track Generated By A Single Array (X vs Z).	5
Figure 5.	Track Generated By Two Arrays In An Overlap Region (X vs Y).	6
Figure 6.	Track Generated By Two Arrays In An Overlap Region (X vs Z).	7
Figure 7.	Kalman Filter Variance Of X Estimate (Small $Q(k)$).	12
Figure 8.	Kalman Filtered Track - X vs. Y (Small $Q(k)$).	13
Figure 9.	Kalman Filtered Track - X vs. Z (Small $Q(k)$).	14
Figure 10.	Kalman Filtered Variance Of X Estimate (Large $Q(k)$).	15
Figure 11.	Kalman Filtered Track - X vs. Y (Large $Q(k)$).	16
Figure 12.	Kalman Filtered Track - X vs. Z (Large $Q(k)$).	17
Figure 13.	Kalman Filtered & Smoothed Variances Of X (Small $Q(k)$).	20
Figure 14.	Raw & Smoothed Tracks - X vs. Y (Small $Q(k)$).	21
Figure 15.	Raw & Smoothed Tracks - X vs. Z (Small $Q(k)$).	22
Figure 16.	Kalman Filtered & Smoothed Variances Of X (Large $Q(k)$).	23
Figure 17.	Raw & Smoothed Tracks - X vs. Y (Large $Q(k)$).	24
Figure 18.	Raw & Smoothed Tracks - X vs. Z (Large $Q(k)$).	25
Figure 19.	Kalman Filtered & Smoothed Variances Of X (Arrays 1 & 11).	27
Figure 20.	Kalman Filtered Track - X vs. Y (Arrays 1 & 11).	28
Figure 21.	Smoothed Track - X vs. Y (Arrays 1 & 11).	29
Figure 22.	Raw & Smoothed Tracks - X vs. Y (Arrays 1 & 11).	30
Figure 23.	Kalman Filtered Track - X vs. Z (Arrays 1 & 11).	31
Figure 24.	Smoothed Track - X vs. Z (Arrays 1 & 11).	32
Figure 25.	Raw & Smoothed Tracks - X vs. Z (Arrays 1 & 11).	33
Figure 26.	Raw Track - X vs. Y (Arrays 2 & 12).	34
Figure 27.	Raw Track - X vs. Z (Arrays 2 & 12).	35
Figure 28.	Kalman Filtered & Smoothed Variances Of X (Arrays 2 & 12).	36
Figure 29.	Kalman Filtered Track - X vs. Y (Arrays 2 & 12).	37
Figure 30.	Smoothed Track - X vs. Y (Arrays 2 & 12).	38
Figure 31.	Raw & Smoothed Tracks - X vs. Y (Arrays 2 & 12).	39

Figure 32. Kalman Filtered Track - X vs. Z (Arrays 2 & 12).	40
Figure 33. Smoothed Track - X vs. Z (Arrays 2 & 12).	41
Figure 34. Raw & Smoothed Tracks - X vs. Z (Arrays 2 & 12).	42

I. INTRODUCTION

The short base-line array tracking ranges located near the Naval Torpedo Station at Keyport, Washington, are used to conduct torpedo testing in support of torpedo development projects. The tracking system in use on these ranges consists of a series of short base-line arrays which can independently track several subsurface targets by relaying received signals from each target to a tracking computer. The computer processes the received array data and calculates target tracks in X,Y,and Z coordinates for display. The arrays are distributed over the range so that full coverage of the range is achieved, and as a result there are several regions on the range where the arrays overlap in their coverage. Figure 1 on page 2 is an illustration of this arrangement [Ref. 1: p. 199]. *Handoff* is a term used to describe when the tracking information on a target supplied to the tracking computer is shifted from one array to a neighboring array.

Each short base-line array consists of four omnidirectional hydrophones spaced 30 feet apart on orthogonal booms as shown in Figure 2 on page 3. The four hydrophones are all connected to a common cable which feeds the received signal from each hydrophone to the tracking computer. The computer extracts the different times of arrival of the received signals and calculates X,Y, and Z coordinates based on the following equations:

$$X = \frac{V^2}{2D} (Tc^2 - Tx^2) \quad (1.1)$$

$$Y = \frac{V^2}{2D} (Tc^2 - Ty^2) \quad (1.2)$$

$$Z = \frac{V^2}{2D} (Tc^2 - Tz^2) \quad (1.3)$$

where:

V = speed of sound in water,

D = hydrophone separation distance,

Tc = tracking signal travel time from target to hydrophone C,

Tx = tracking signal travel time from target to hydrophone X,

Ty = tracking signal travel time from target to hydrophone Y, and

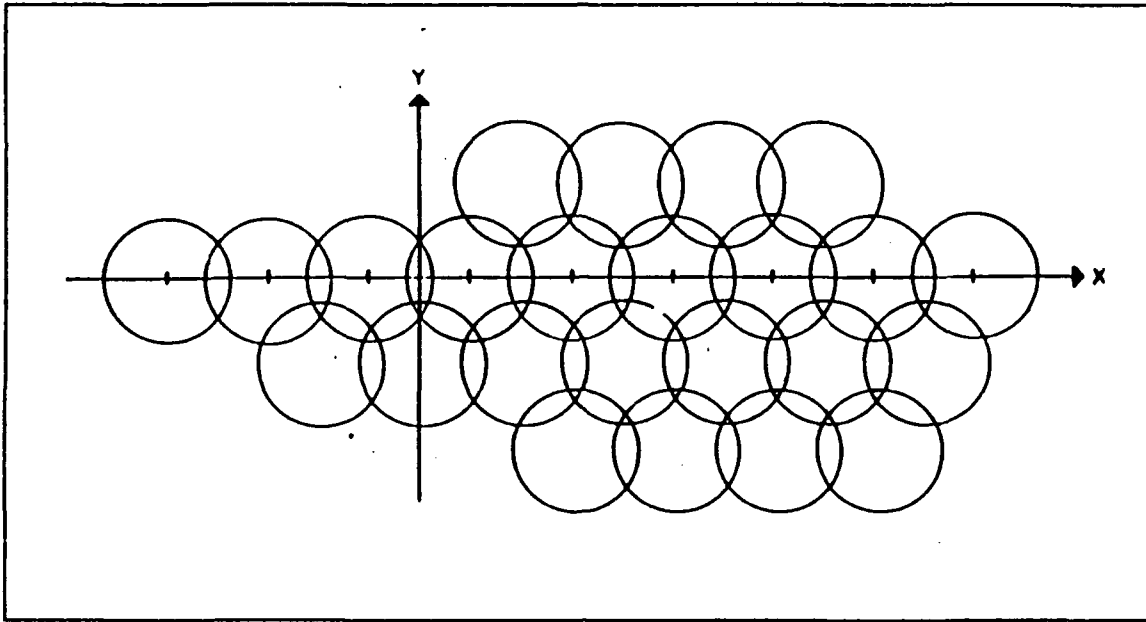


Figure 1. Short Base-line Array Range Configuration - Fig 2 From Ref 1.

T_z = tracking signal travel time from target to hydrophone Z.

A more detailed description of the array and derivation of the equations above can be found in [Ref. 2].

The tracking signal used on these ranges is a 150 watt phase shift keying (PSK) pulse consisting of 47 bits (19 identification bits and 27 telemetry bits) and lasting approximately five milliseconds. The 0's and 1's making up the bit stream are 180 degrees out of phase. Correlation processing techniques are used to validate the received signal, thus improving the signal to noise ratio. [Ref. 3: p. 9]

Figures 3 and 4 show examples of a typical track generated by this system. Figure 3 on page 4 is the track in X vs. Y coordinates generated from data received by a single array while Figure 4 on page 5 shows the same tracks in X vs. Z coordinates. It is obvious that these tracks are affected by random noise. Figures 5 and 6 show examples of a track generated with data from two arrays as would occur in an overlap region. Again, the X vs. Y coordinates and X vs. Z coordinates format is used. It can be seen that the data from the different arrays do not agree. This indicates that bias errors are also present in the current tracking system. Tracking accuracy has been improved by use of the PSK pulse because its sophistication increases the received signal

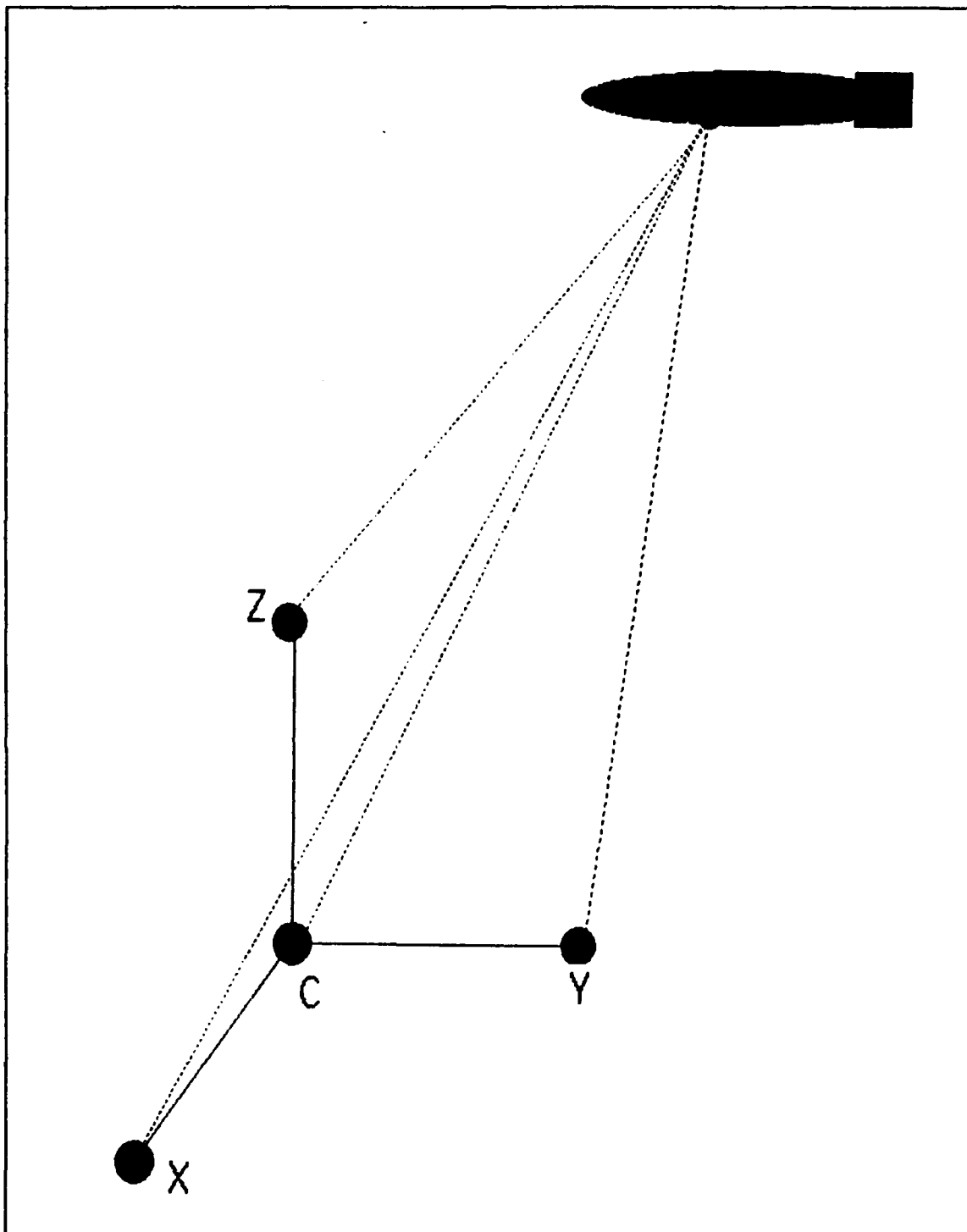


Figure 2. Short Base-line Array Configuration.

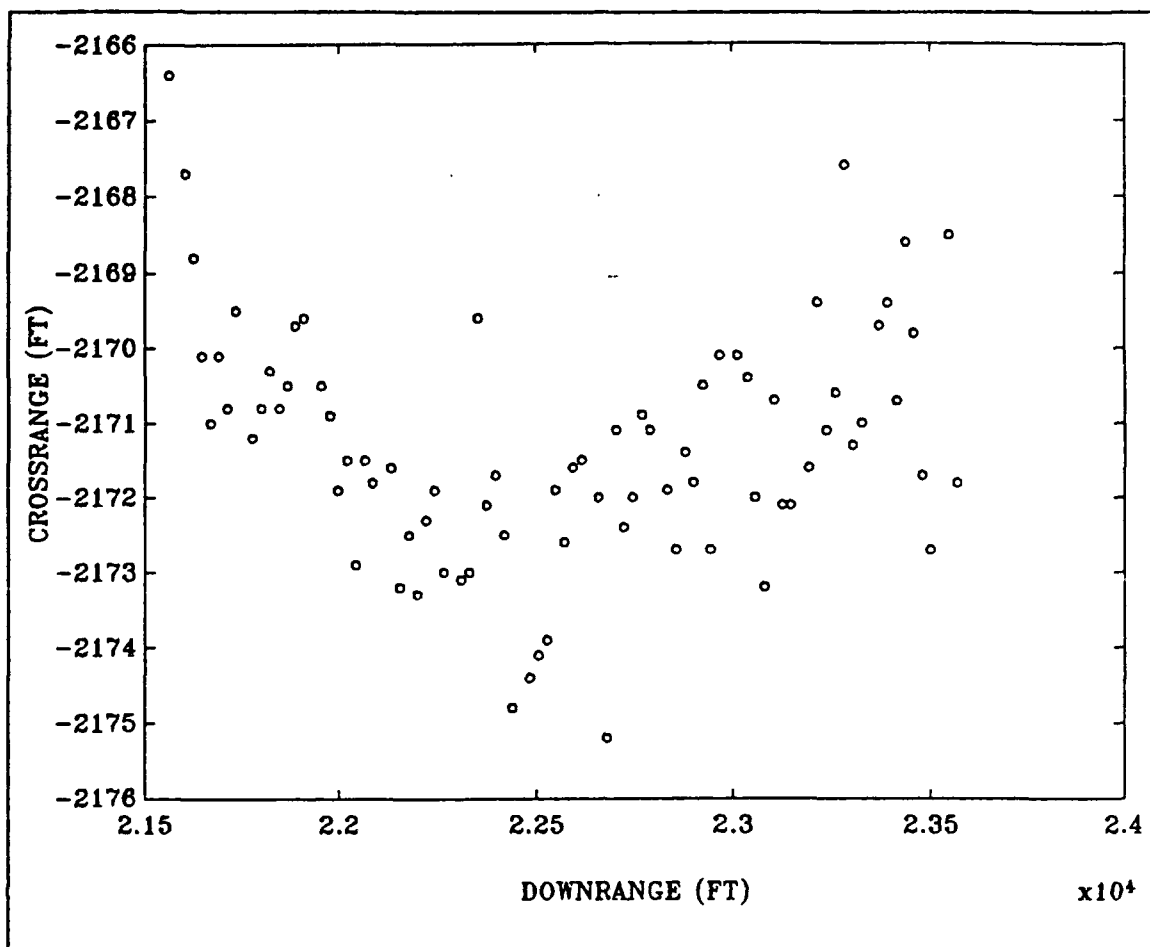


Figure 3. Typical Track Generated By A Single Array (X vs Y).

to noise ratio, but it can be seen from the tracks that a great deal of noise remains. The question that now arises is how these target tracks can be improved.

While several approaches to improving the tracking accuracy of the short base-line array ranges are possible, the effort here has been on applying a Kalman Filter and Fixed Interval Smoothing Algorithm as a post data analysis tool. Quite a bit of work has already been done in the area of applying Kalman Filters to underwater tracking over the years [Refs. 4, 5, 6, 7]. More recent efforts have centered around applying an optimal smoothing algorithm to the underwater tracking problem [Refs. 8, 9]. Most of this work has involved attempts to filter or smooth the transit times prior to their use in the tracking equations using an Extended Kalman Filter. In the program developed in this research, a Kalman Filter utilizing a second order linear model is used to reduce

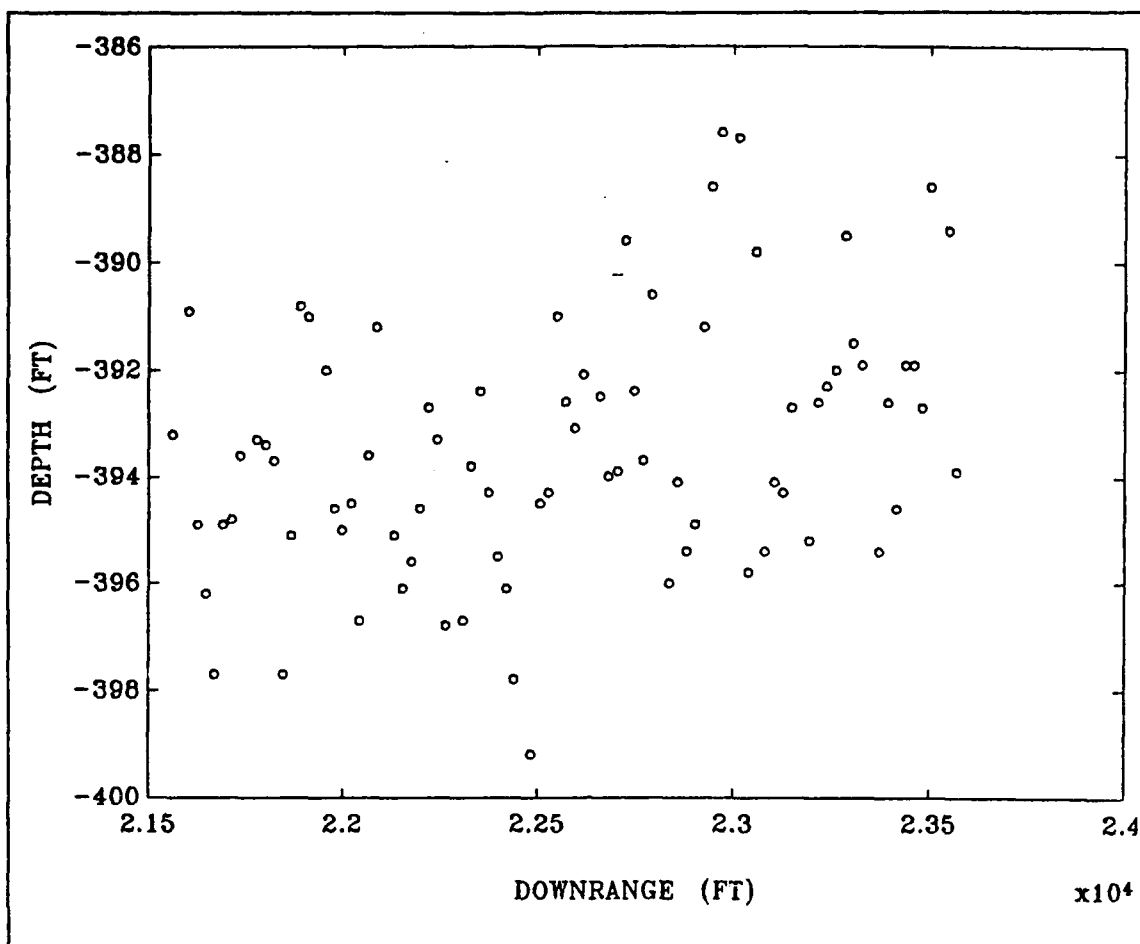


Figure 4. Typical Track Generated By A Single Array (X vs Z).

the random noise in the already calculated X, Y, and Z position coordinates. The optimal Fixed Interval Smoothing algorithm is then used to improve the track quality even further. This method is used because it lends itself more readily to post data analysis. The program is written in Fortran, compiled using the Microsoft, Inc. 4.01 Optimal Compiler and run on an IBM-AT personal computer. Details of the program and results obtained will now be discussed.

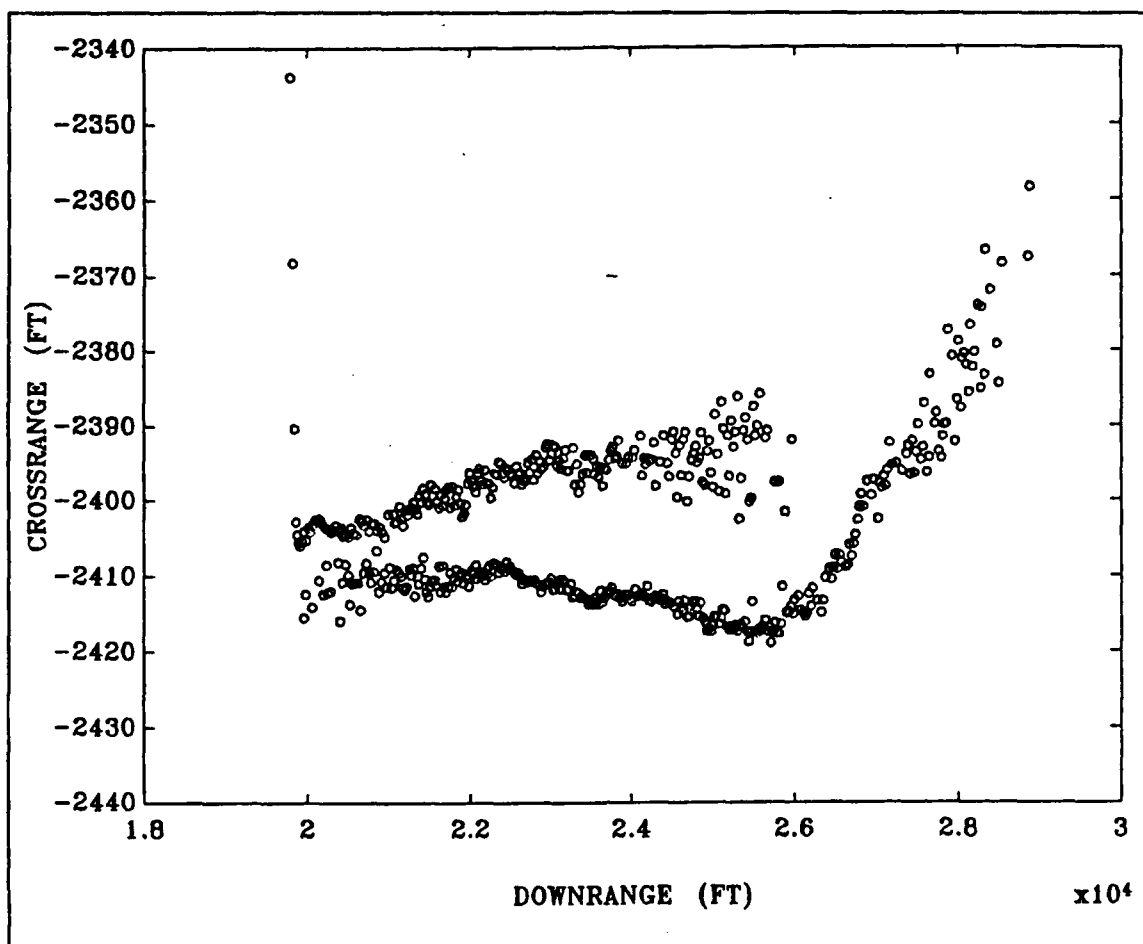


Figure 5. Track Generated By Two Arrays In An Overlap Region (X vs Y).

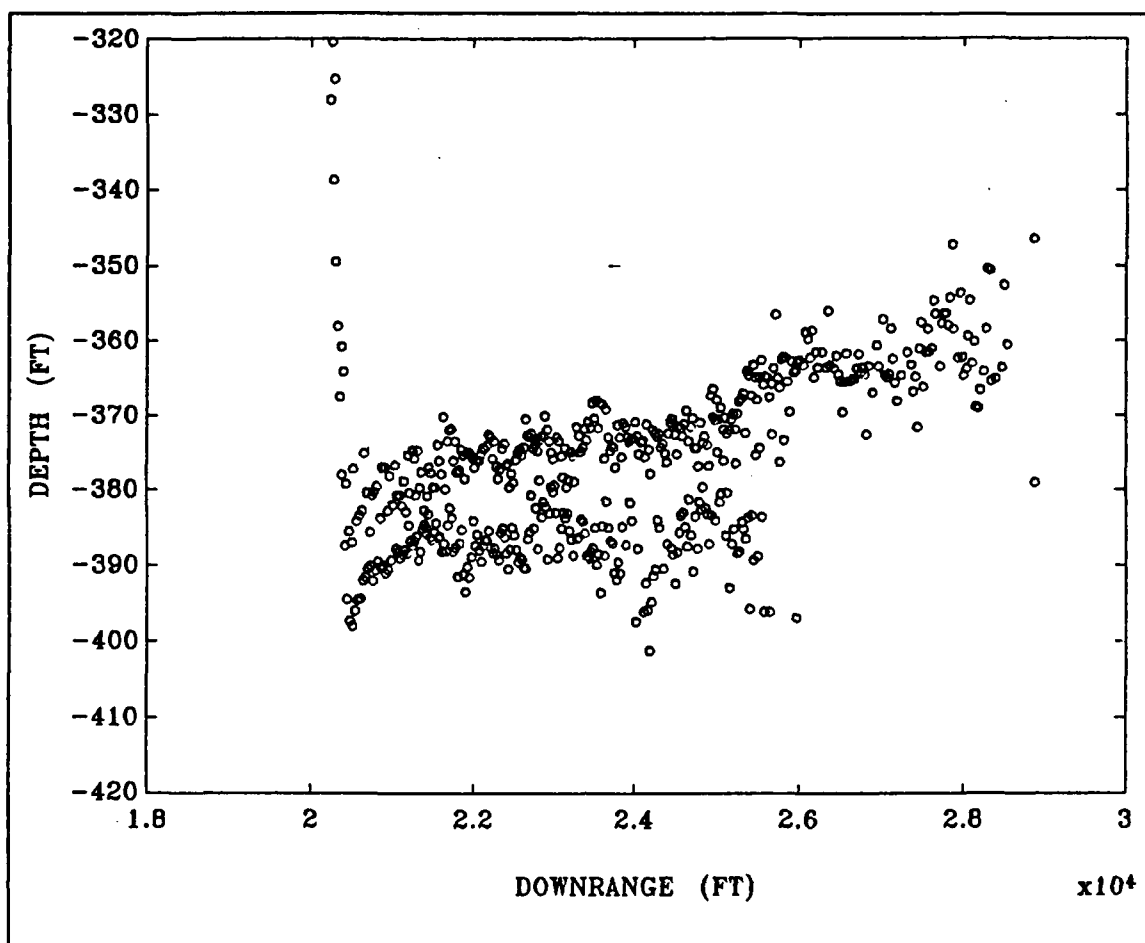


Figure 6. Track Generated By Two Arrays In An Overlap Region (X vs Z).

II. KALMAN FILTER

The Kalman Filter is designed to remove random noise from the estimate of the parameter with which it is concerned by adding a weighted error term to the observed parameter being estimated. The error term is simply the difference between the filter's prediction of the parameter and the actual value observed at a particular time. The weighting factor is influenced by the magnitude of the measurement error and the covariance of error between the estimate and the observation. The covariance of error of the estimate is also updated based on previous values of the covariance of error. Many treatments of the derivation and application of Kalman Filters were found helpful in preparing this report [Refs. 10, 11, and 12], and the equations used in this application will now be presented.

The equations shown below were obtained directly from previous work on this subject [Ref. 9: p. 19]. Their development will not be repeated here as an excellent derivation of these equations is presented in [Ref. 13: pp. 176-182].

$$\hat{\mathbf{x}}(k | k) = \hat{\mathbf{x}}(k | k - 1) + G(k)[\mathbf{z}(k) - \hat{\mathbf{z}}(k | k - 1)] \quad (2.1)$$

$$\hat{\mathbf{x}}(k + 1 | k) = \phi \hat{\mathbf{x}}(k | k) \quad (2.2)$$

$$\hat{\mathbf{z}}(k | k - 1) = H \hat{\mathbf{x}}(k | k - 1) \quad (2.3)$$

$$G(k) = P(k | k - 1) H^T [H P(k | k - 1) H^T + R]^{-1} \quad (2.4)$$

$$P(k + 1 | k) = \phi P(k | k) \phi^T + Q(k) \quad (2.5)$$

$$P(k | k) = [I - G(k) H] P(k | k - 1) \quad (2.6)$$

where:

$\hat{\mathbf{x}}$ - state estimate vector,

$\hat{\mathbf{z}}$ - observation vector,

H - measurement matrix,

ϕ - state transition matrix,

G - Kalman gain matrix,

P - covariance of estimate error matrix,

R - covariance of measurement error matrix,

Q - covariance of excitation error matrix, and

I - identity matrix.

In this application, six states were used; namely, the x , y , and z positions and the corresponding velocities, \dot{x} , \dot{y} , and \dot{z} . The model for the system, represented by the state transition matrix ϕ , is a second order linear model

$$\phi = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

which when multiplied by the state matrix $\hat{x}(k | k)$ results in the equations of motion

$$\hat{x}(k+1) = \hat{x}(k) + \hat{\dot{x}}(k)T \quad (2.8)$$

$$\hat{y}(k+1) = \hat{y}(k) + \hat{\dot{y}}(k)T \quad (2.9)$$

$$\hat{z}(k+1) = \hat{z}(k) + \hat{\dot{z}}(k)T \quad (2.10)$$

where T represents the time in seconds between samples. Since the parameters being estimated are already expressed in a linear coordinate system, the measurement matrix H necessary in this case need only extract the observable states, i.e., x , y , and z . The measurement matrix used is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.11)$$

The values of $Q(k)$ and R chosen play an important role in the performance of the filter. The covariance of excitation error, $Q(k)$, is a measure of how confident the filter is in the adequacy of the system model together with how strong the noise affecting the

actual system is expected to be. A relatively large value of $Q(k)$ results in the filter placing more emphasis on the observation and less on the predicted value when updating the estimate, while a smaller value of $Q(k)$ has the opposite effect. Therefore, with a larger $Q(k)$, the filter output should be noisier but better able to handle disturbances not related to noise such as, in this case, actual target maneuvers. On the other hand, the covariance of measurement error R indicates the confidence in the accuracy of the data measurements made. It turns out that increasing the magnitude of R in effect decreases the gain resulting in less weight being given to the difference between the predicted estimate and the actually observed value. This results in less attention being paid to the noise observed, which makes sense if it is assumed that the measured values are less accurate. [Ref. 12: p. 224]

The covariance of excitation error matrix used for this filter application is as follows:

$$Q = \begin{bmatrix} \frac{T^4}{4} W & 0 & 0 & 0 & 0 & 0 \\ 0 & T^2 W & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{T^4}{4} W & 0 & 0 & 0 \\ 0 & 0 & 0 & T^2 W & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{T^4}{4} W & 0 \\ 0 & 0 & 0 & 0 & 0 & T^2 W \end{bmatrix} \quad (2.12)$$

where:

$$W = \sigma_x^2 = \sigma_y^2 = \sigma_z^2 = E[ww^T] \quad (2.13)$$

A detailed derivation of this matrix is shown in [Ref. 4: pp. 36-42]. Since the excitation errors are assumed to be uncorrelated, only the diagonal terms of this matrix are non-zero. The value of W is set by the user while running the program. The value of R chosen is 25 square feet. This is a conservative value based on range accuracy estimates reported by NUWES [Ref. 3: p. 6].

Finally, in initializing the filter, an initial value for the covariance of estimate error $P(k/k)$ must be chosen. This value affects the transient response of the filter early on, but does not affect the steady state response [Ref. 12: p. 224]. For this filter, $P(k/k)$ initially is set at one million square feet. Again, uncorrelated errors were assumed, so, only the main diagonal terms are non-zero. The initial value of $P(k/k)$ used here was arrived at mainly through trial and error.

In summary, the Kalman Filter is a linear minimum variance estimator whose output is nothing more than the conditional mean of the parameter being estimated based on the observations made. A more detailed description of the actual programs can be found in Appendix A and Appendix B.

This program was run on a typical data file obtained from information provided by the Keyport Range. Figures 3 and 4 show the raw track in X vs. Y and X vs. Z coordinates respectively. Figure 7 on page 12 shows the Kalman Filter variance of the estimate of the X coordinate of the target's track. It can be seen that the variance settles quickly to a steady state value of about five square feet. The added uncertainty, introduced by the fact that periodically samples are missed by the range tracking system, is evident in that the values of $P(k/k)$ do not settle to a constant value, but this uncertainty is absorbed by the filter and causes no problem in the filter's performance.

Figure 8 on page 13 shows the Kalman filtered track for the coordinates X vs. Y. Clearly, the filtered track is much smoother than the raw data and follows the raw track closely after the filter settles out. However, the filter tracks a little low after the target appears to make a sharp maneuver to its left because the value of $Q(k)$ is small. In other words, a small value of $Q(k)$ does not allow the filter to follow maneuvers well, and this raw data set looks like it is maneuvering initially. Figure 9 on page 14 is the same data as Figure 8 on page 13 except that the Z coordinates are plotted against the X coordinates. This plot does not exhibit the same track off behavior because no maneuvers are apparent and, in fact, a great deal of noise reduction is shown.

Another run of the program was made using a value of $Q(k)$ increased by a factor of 100. Figure 10 on page 15 shows the variance of x and, as expected, it follows the same pattern but settles at a higher "steady state" value of approximately 11 square feet. Also, the jumps in $P(k/k)$ due to the missed samples are more pronounced. Figure 11 on page 16 is a plot of X vs. Y and, when compared to Figure 8 on page 13, clearly exhibits the expected behavior. The filtered track is much noisier, but it does not track off as before. Figure 12 on page 17 displays X vs. Z and again is a noisier track which follows the raw data more closely.

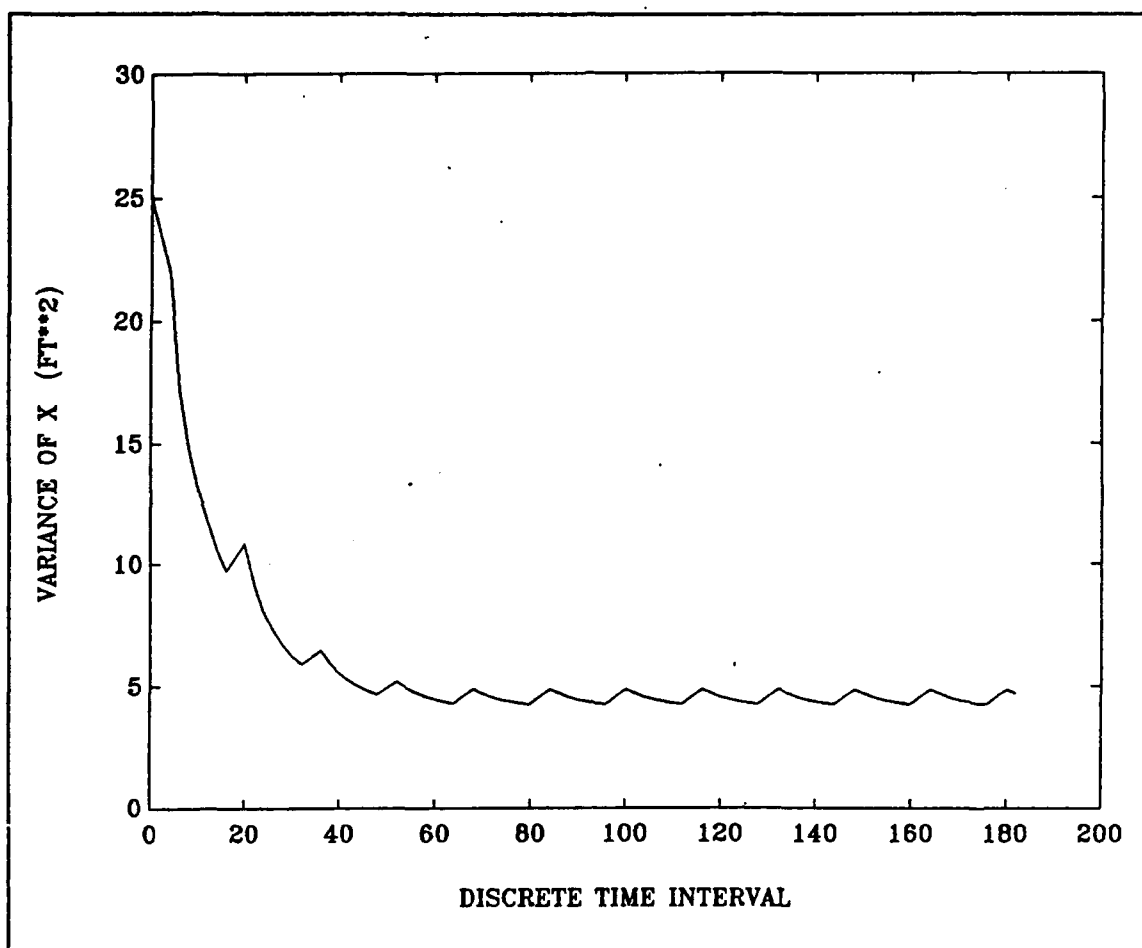


Figure 7. Kalman Filter Variance Of X Estimate (Small $Q(k)$).

The Kalman Filter routine designed for this application provides the user with a filtered set of data points which are an improvement over the actual raw data available in that, as expected, much of the random noise which tends to corrupt the raw data has been eliminated. Proper selection of the R and $Q(k)$ matrices and the initial value of $P(k/k)$ ensures that the filter is receptive to target maneuvers while at the same time significantly reducing the random noise. The user must decide on the proper balance to fit his needs based on known information such as the expected maneuverability of the target and the magnitude of the expected measurement errors. In addition, this filtered data is now ready to be processed by the Fixed Interval Smoothing Routine.

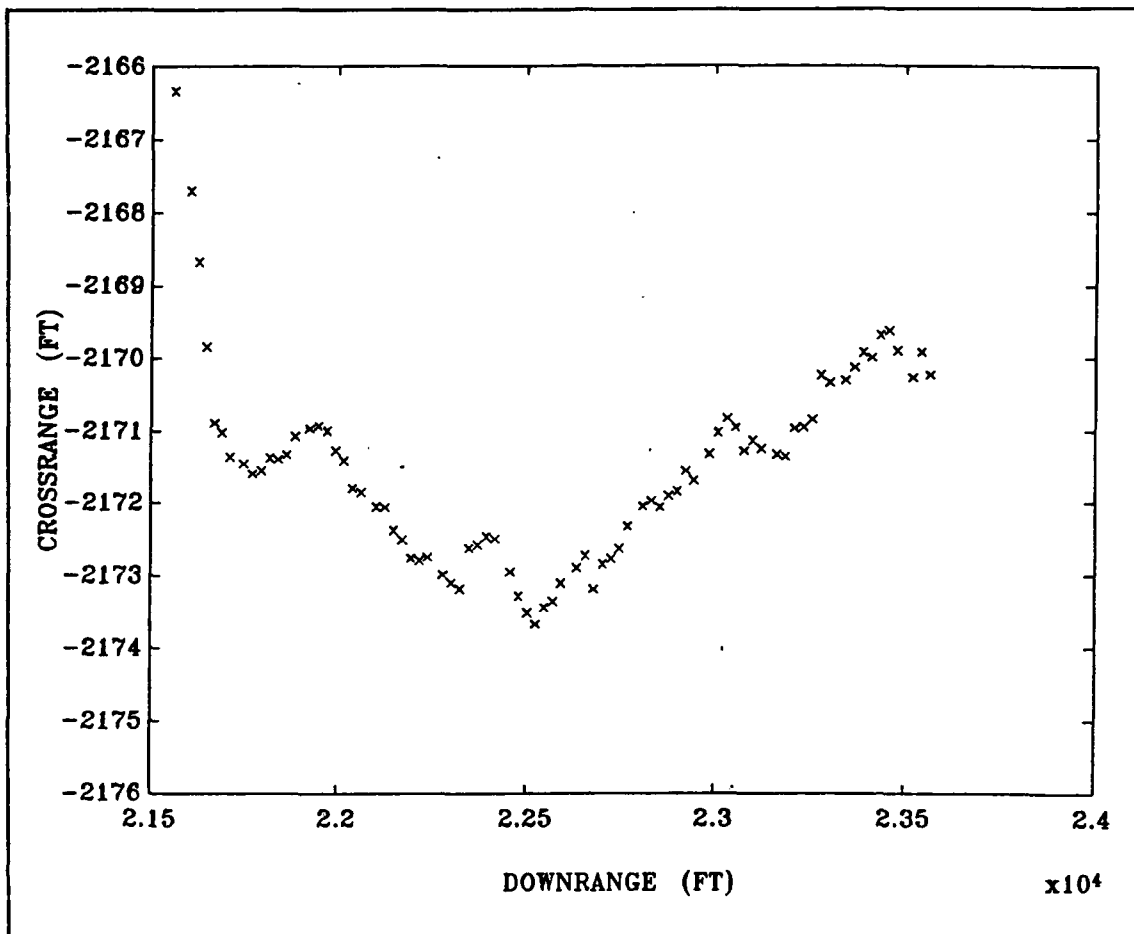


Figure 8. Kalman Filtered Track - X vs. Y (Small $Q(k)$).

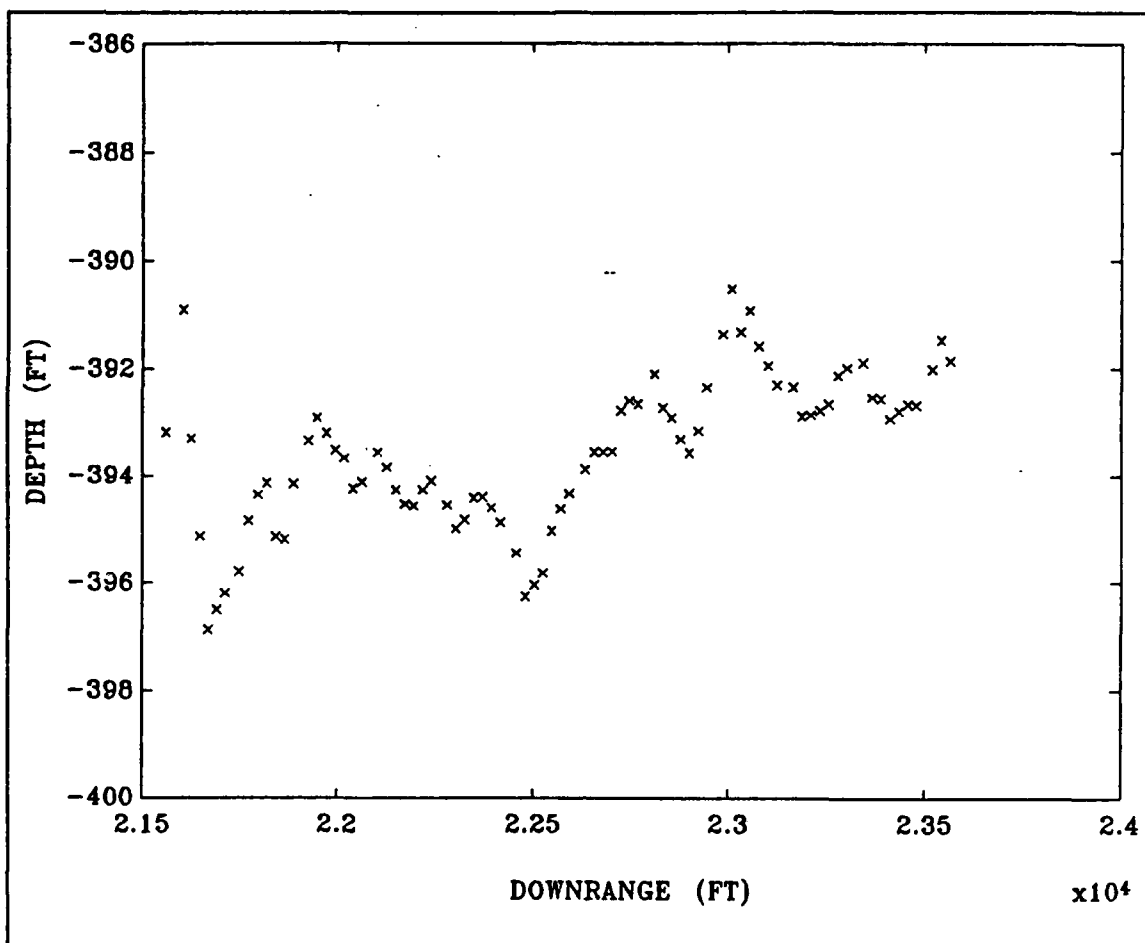


Figure 9. Kalman Filtered Track - X vs. Z (Small $Q(k)$).

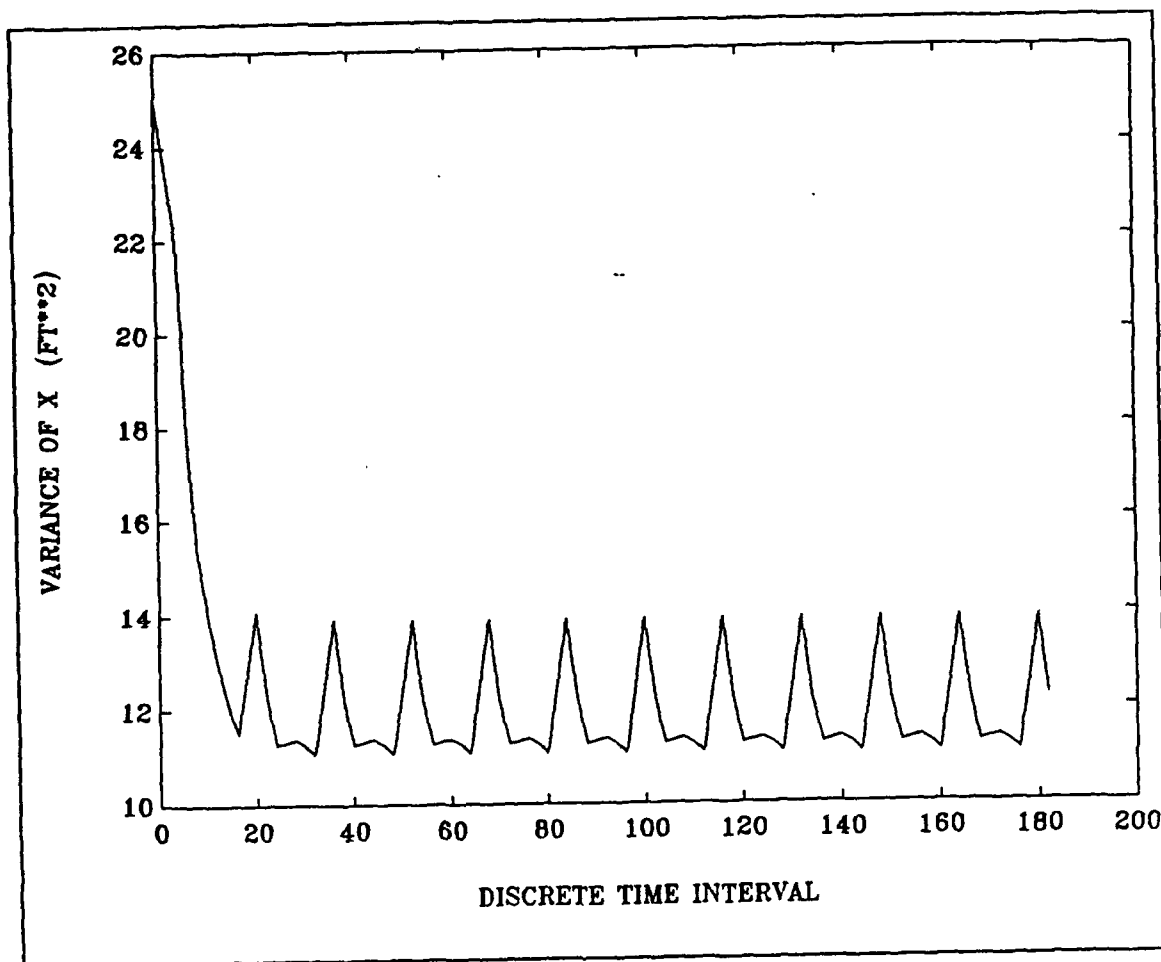


Figure 10. Kalman Filtered Variance Of X Estimate (Large $Q(k)$).

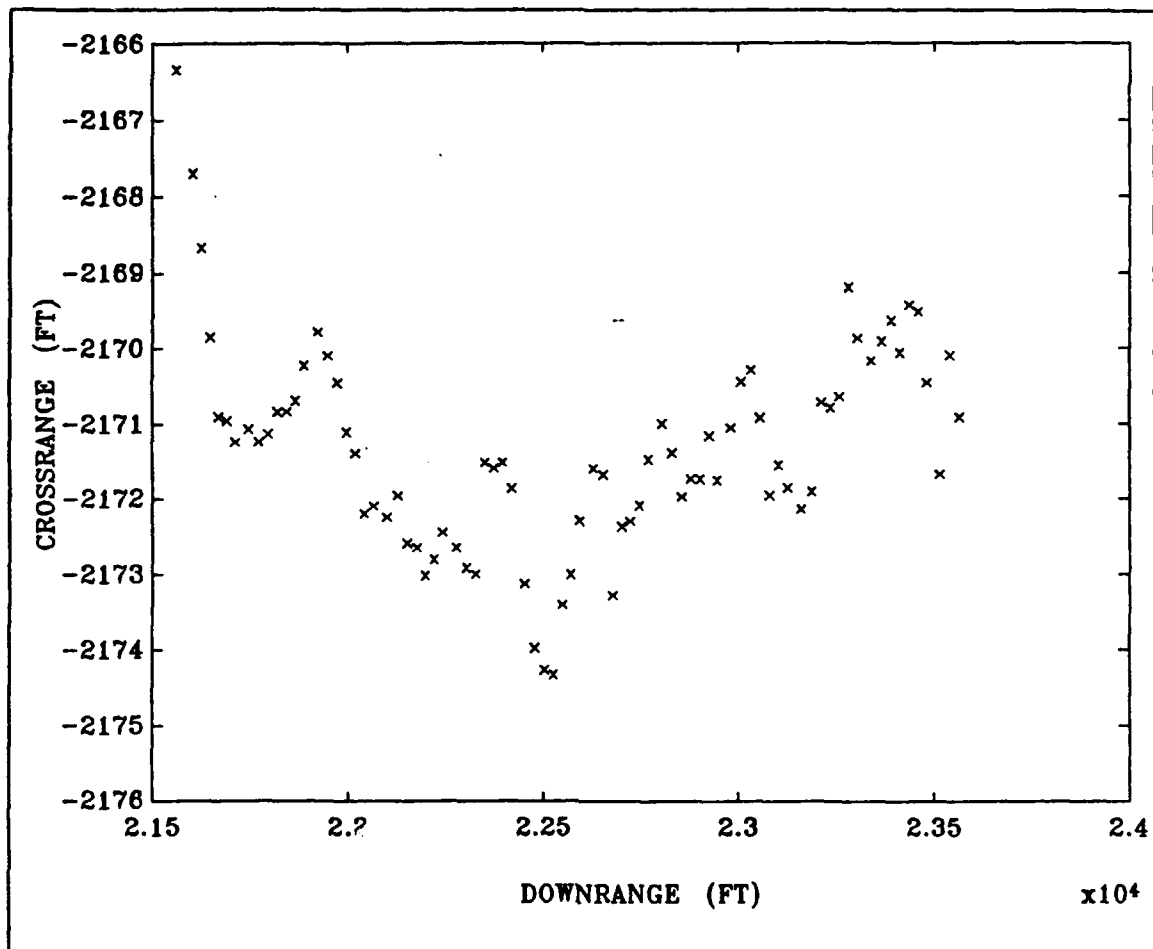


Figure 11. Kalman Filtered Track - X vs. Y (Large $Q(k)$).

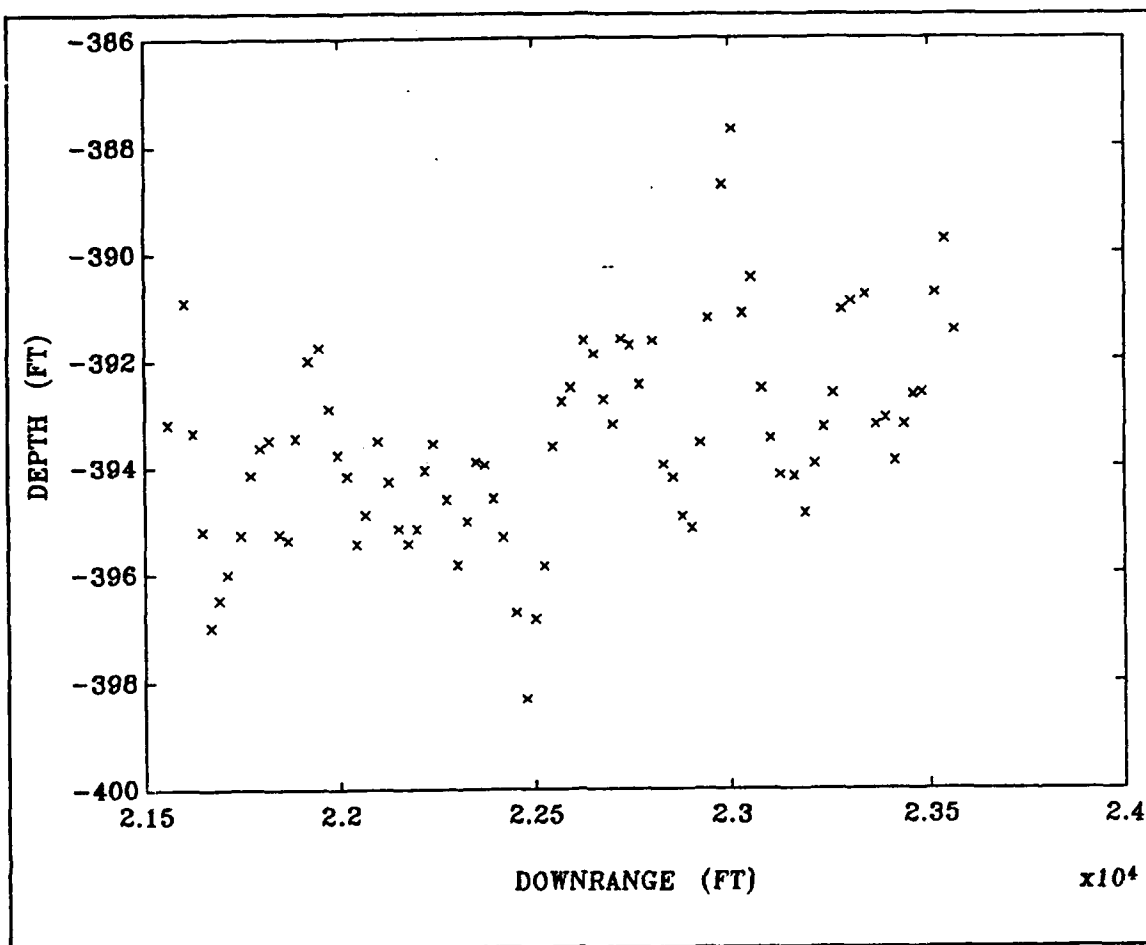


Figure 12. Kalman Filtered Track - X vs. Z (Large $Q(k)$).

III. FIXED INTERVAL SMOOTHING ALGORITHM.

The Fixed Interval Smoothing Algorithm is one of the three algorithms designed to improve upon the Kalman Filter's results by taking into account data which was not available when each Kalman Filter estimate was made and updating each previous estimate accordingly. The Fixed Point and Fixed Lag Smoothing algorithms, while very similar to the Fixed Interval Smoothing routine, differ in the way they include this additional data into each estimate. The Fixed Interval Smoothing algorithm recalculates each estimate generated by its associated Kalman Filter based on information obtained over the entire interval of data being analyzed. In this sense, it is useful only as a post data analysis tool, since the entire set of data over the given interval must be known, and Kalman Filter estimates and covariance of error between estimates and observations must be generated previously. In implementing this algorithm, a system of recursive equations which operate backwards in time from the last data point to the first data point are used.

The equations for the Fixed Interval Smoothing algorithm used in this application were obtained from [Ref. 9: p. 20]. Several sources were beneficial in understanding these equations [Refs. 10, 11, and 14] and [Ref. 13: pp. 216-225] provides an excellent derivation of them. Therefore, these derivations will not be repeated here. The equations of interest are:

$$\hat{\mathbf{x}}(k | N) = \hat{\mathbf{x}}(k | k) + A(k)[\hat{\mathbf{x}}(k+1 | N) - \hat{\mathbf{x}}(k+1 | k)] \quad (3.1)$$

$$A(k) = P(k | k)\phi^T P(k+1 | k)^{-1} \quad (3.2)$$

$$P(k | N) = P(k | k) + A(k)[P(k+1 | N) - P(k+1 | k)]A(k)^T \quad (3.3)$$

where:

- $\hat{\mathbf{x}}(k | N)$ - smoothed estimate at sample k ,
- $P(k | N)$ - smoothed covariance of estimate error at sample k ,
- $A(k)$ - smoothing algorithm gain matrix, and
- $\hat{\mathbf{x}}(k | k)$ and $P(k | k)$ - values stored by the Kalman Filter routine.

It can be seen from the above equations that the estimate generated by the Fixed Interval Smoothing algorithm is simply the Kalman Filter estimate adjusted by a

weighted error term. The error term is the difference between the smoothed estimate calculated for the previous data point (which is actually the next sequential data point in the file in real time), and the predicted value of the corresponding parameter generated by the Kalman Filter. The gain matrix $A(k)$ is dependent on the covariance of error between estimate and observation generated by the Kalman Filter, namely, $P(k/k)$ and $P(k+1/k)$. The smoothed value of the covariance of error $P(k/N)$, while having no impact on the smoothed estimate $\hat{x}(k | N)$, provides a measure of how well the smoothing filter is working. In general, $P(k/N)$ should be less than $P(k/k)$ except at the N th point where they should equal one another. As the smoothing filter moves backward in time, it adjusts the original Kalman Filter estimate depending on the smoothed estimate's agreement with the predicted value for the previous point operated on, and on the confidence level of the Kalman Filter in its own solution as indicated by the values of $P(k/k)$ and $P(k+1/k)$. Simply stated, if more uncertainty exists in the Kalman Filter Solution, more weight is given to the difference between previous smoothed estimates and predicted values of the parameter in computing the current smoothed estimate.

The Fixed Interval Smoothing portion of the program listed in Appendix A implements the equations shown above. As with the Kalman Filter portion of the program described in the previous chapter, there is no provision for the resultant data to be displayed graphically by this program itself because the size of the data files involved exceeds the capabilities of the plotting routines available. However, graphical results are easily obtained from the data files output by the program using Matlab. Details of Matlab plotting capabilities and descriptions of the commands used to generate the plots shown can be found in [Ref. 15].

Results of the smoothing algorithm were obtained using the Kalman Filter results presented in the previous chapter as inputs to the smoothing loop of the program. As in chapter 2, data is presented in the X vs. Y plane and the X vs. Z plane, and cases were run for both values of $Q(k)$.

Figure 13 on page 20 shows the variance of the x estimate associated with the smoothing routine together with the variance of x achieved with the Kalman Filter. As expected, the value of the smoothed variance is a definite improvement over the Kalman Filter result, and in fact settles to an average value that is less than 1.5 square feet. Figure 14 on page 21 is the smoothed version of Figure 8 superimposed on the raw X vs. Y track for the case where $Q(k)$ is small and shows a smooth track which follows the raw track closely. The divergence seen initially in the filtered results for this case has been compensated for, however, a close look at the smoothed results shows some di-

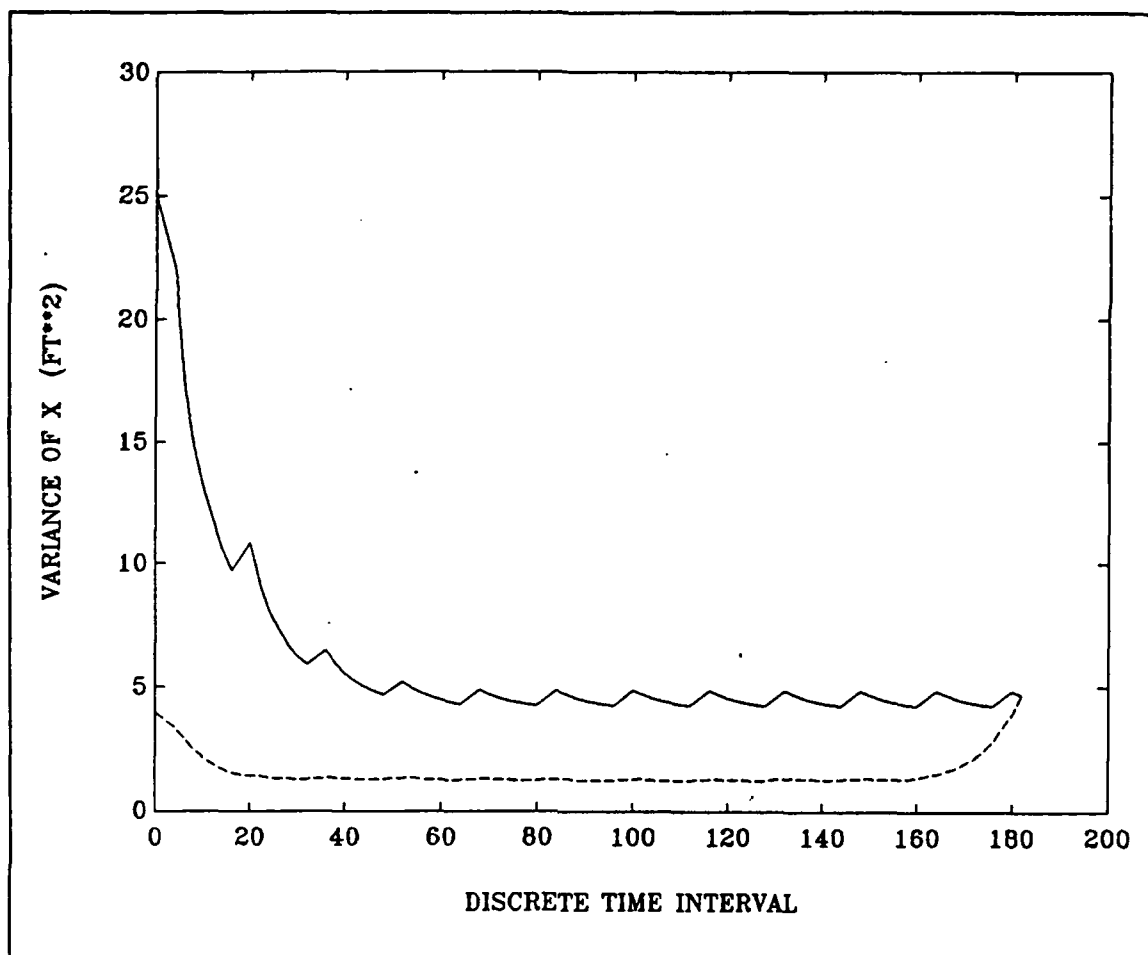


Figure 13. Kalman Filtered & Smoothed Variances Of X (Small $Q(k)$).

vergence still present in the first portion of the track. Figure 15 on page 22 is the smoothed version of Figure 9 superimposed on its raw data, and again a significant improvement is demonstrated.

The Kalman Filter results of the case where a large $Q(k)$ was used were also treated with the smoothing algorithm. Figure 16 on page 23 is the smoothed plot of the variance of the x estimate together with the variance of x shown in Figure 10. This plot indicates that the output is much more influenced by noise and sample uncertainty due to the higher sensitivity when compared to the small $Q(k)$ case, but is a great improvement over the Kalman Filter output. Here the variance settles to an average value of approximately five square feet. Figure 17 on page 24 shows the smoothed version of Figure 11 superimposed on its raw track. This smoothed track greatly improves on the

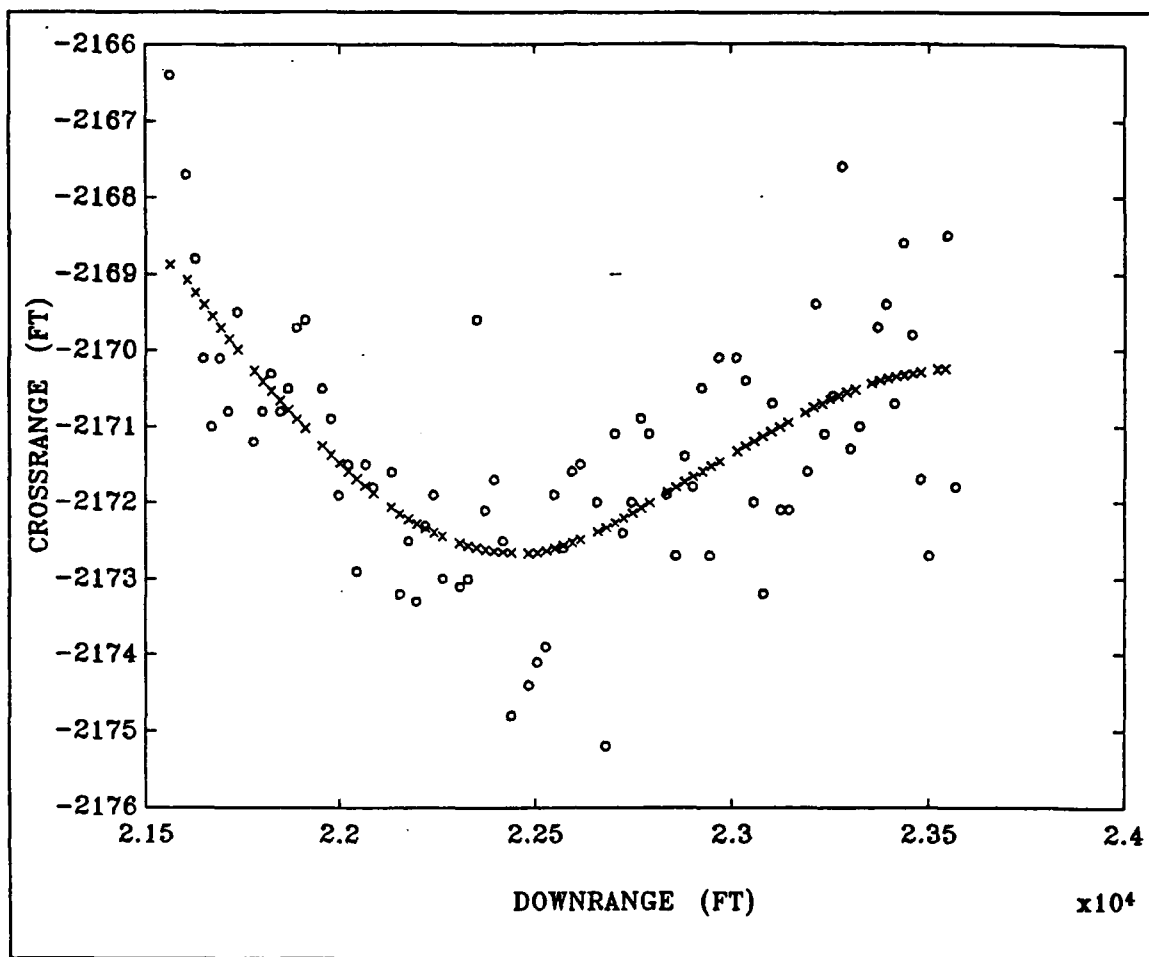


Figure 14. Raw & Smoothed Tracks - X vs. Y (Small $Q(k)$).

filtered track while showing none of the divergence from the raw track which was seen in the small $Q(k)$ case. Figure 18 on page 25 is the raw and smoothed X vs. Z tracks for the large $Q(k)$ case and, when compared to the Kalman Filter results shown in Figure 12, clearly illustrates the excellent performance of the smoothing routine.

These results show that the smoothing algorithm dramatically improves upon the results obtained with the Kalman Filter alone. This intuitively makes sense because the estimates are now based on the behavior of the entire data set instead of on just what is known about the data at the time the estimate is made. The smoothed results depend indirectly upon the values chosen for $Q(k)$ and $P(k/k)$ initially and can even compensate somewhat for poor choices of the parameters in the initialization of the Kalman Filter.

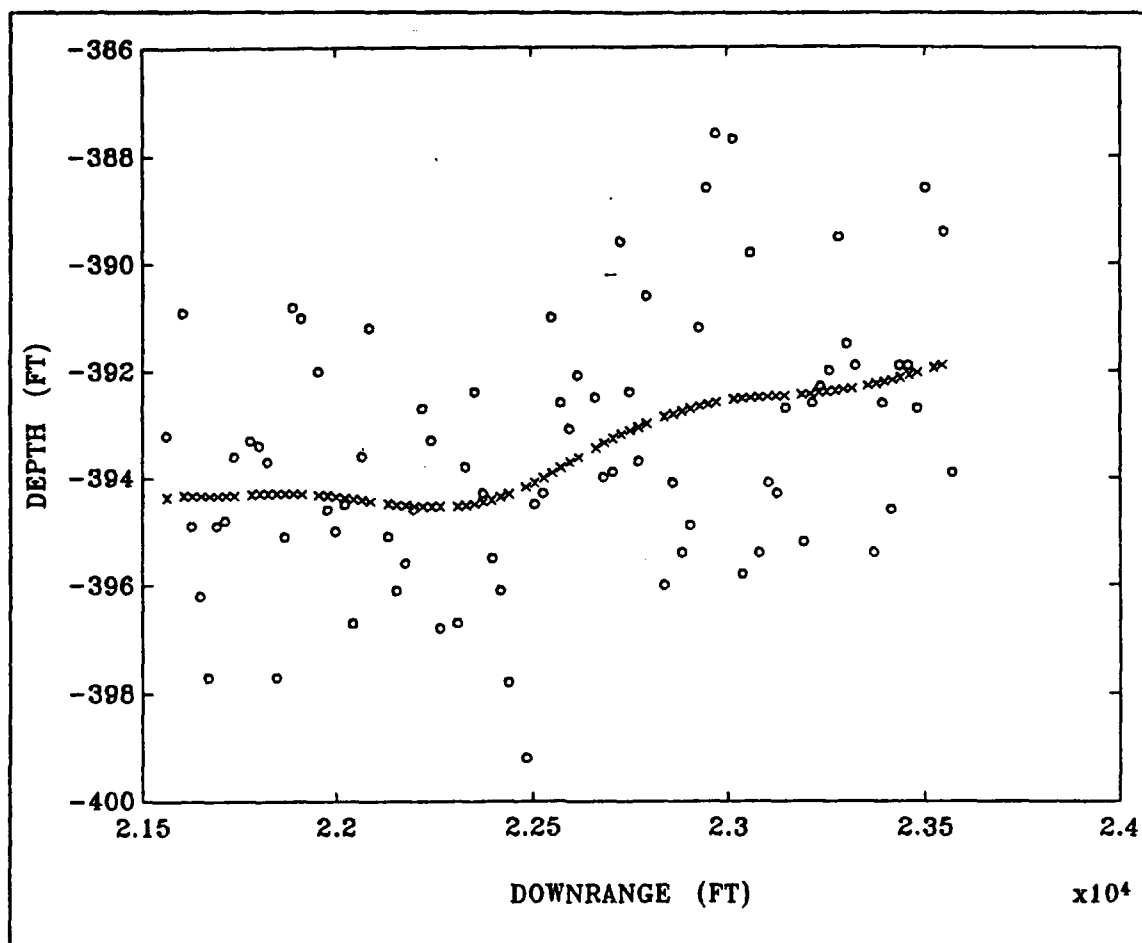


Figure 15. Raw & Smoothed Tracks - X vs. Z (Small $Q(k)$).

These results demonstrate that the Kalman Filter Smoothing Algorithm employed here is an excellent post-data analysis tool.

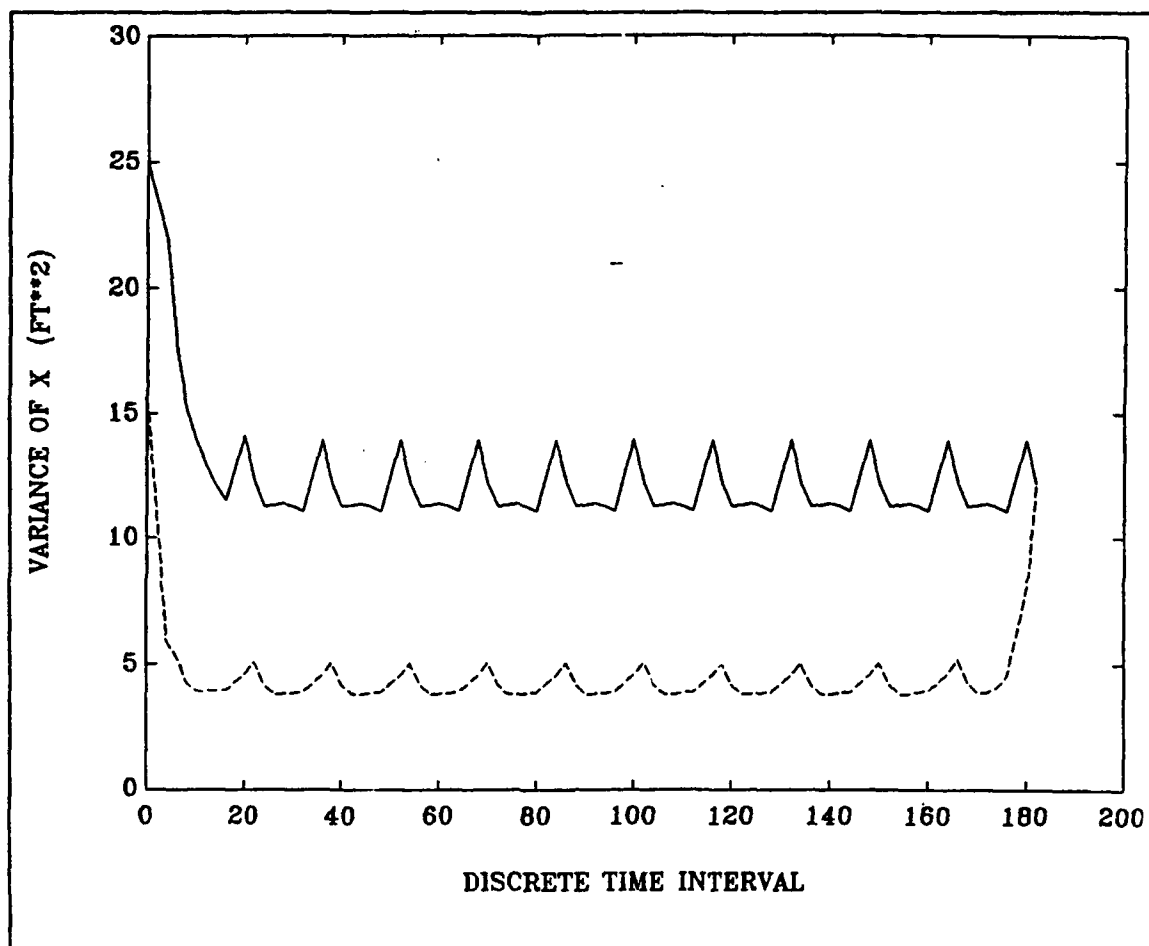


Figure 16. Kalman Filtered & Smoothed Variances Of X (Large $Q(k)$).

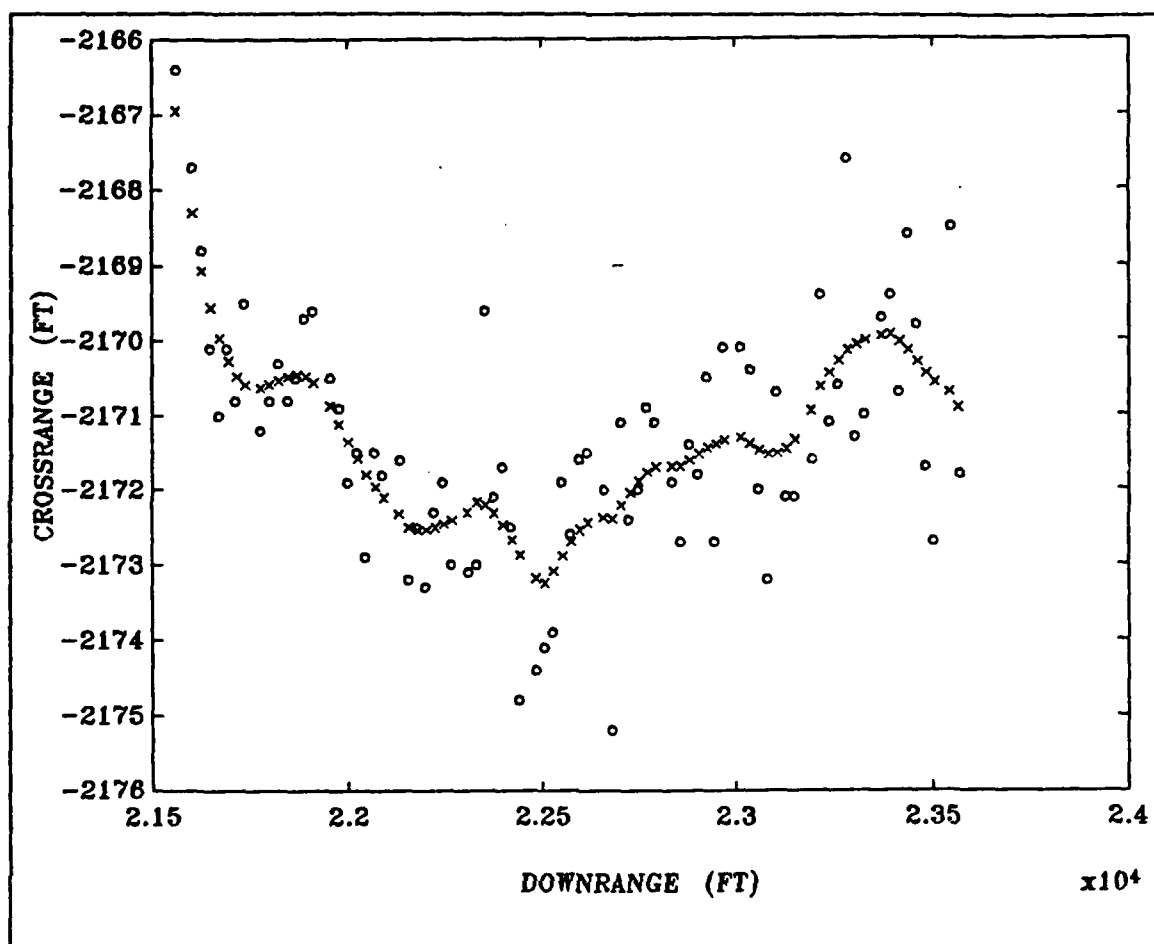


Figure 17. Raw & Smoothed Tracks - X vs. Y (Large $Q(k)$).

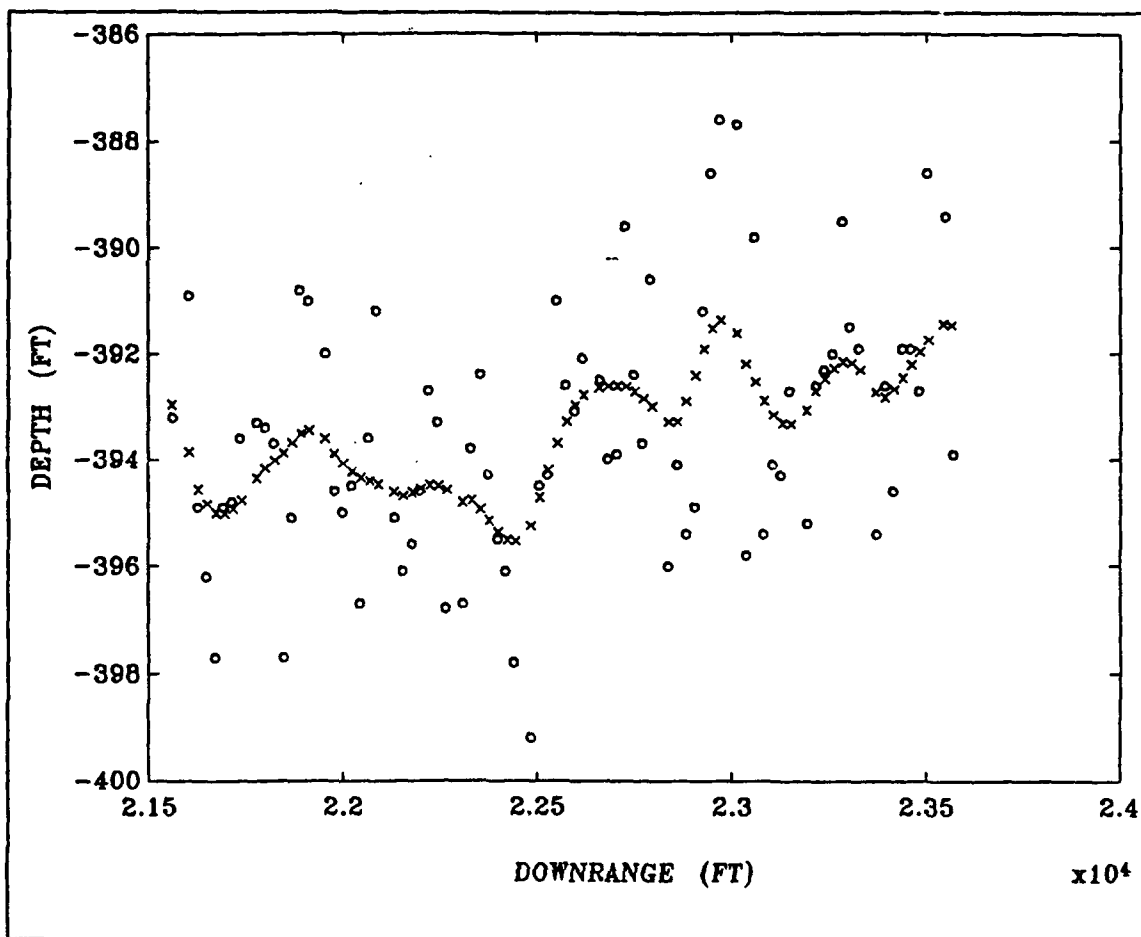


Figure 18. Raw & Smoothed Tracks - X vs. Z (Large $Q(k)$).

IV. SOLVING THE ARRAY HANDOFF PROBLEM.

As discussed in the introduction, one of the major problems with the tracks generated by the current Keyport short base-line tracking array system is the presence of discontinuities in array overlap regions. These discontinuities in the target track caused by differing bias errors present in each array's solution complicate the post data analysis so crucial to torpedo testing and evaluation. It can now be demonstrated that the Fixed Interval Smoothing Kalman Filter Algorithm, as implemented by the program presented here, can improve the overall track quality immensely in the overlap regions.

As shown in previous chapters, the Kalman Filter and Fixed Interval Smoothing routines effectively remove random noise from the generated track data and generate smooth tracks which are easy to see. The processing of track data from two arrays where many sample times have two different values simultaneously is handled quite satisfactorily by the algorithms. The data is treated simply as two distinct samples where no time has elapsed between samples. This overlap data is also characterized by the fact that many samples are missed and occasionally relatively long periods of time pass between data points. It will be seen that this problem is also handled adequately by the algorithms. Another problem experienced with these data sets was that they were too large to be handled by the personal computer's compiler. This problem was solved by switching to the Microsoft, Inc. 4.01 Optimizing Compiler. This compiler not only compiled the program using less memory, but use of the SLARGE metacommand allowed the larger data sets to be processed. Detailed descriptions of this compiler and the available memory models can be found in [Ref. 16]. Two sets of overlap data were analyzed and the results of this analysis are now presented.

Figure 19 on page 27 is a plot of the Kalman Filter variance of the x coordinate estimate vs. time, together with the smoothed variance of the x estimate for the first set of overlap data processed. It is seen that the filtered variance drops quickly to a steady state value of about 6.5 square feet, and then jumps back up erratically over the last third of the set. The thick portion of the filter's variance is where overlap is occurring because here two values of the parameter are available for each time, so, two steady state values of the variance of the estimate are consistently reached. The erratic values of the filter's variance result from the fact that over this region the sample interval varies erratically as samples were missed, so, the uncertainty of the filter is also affected. The

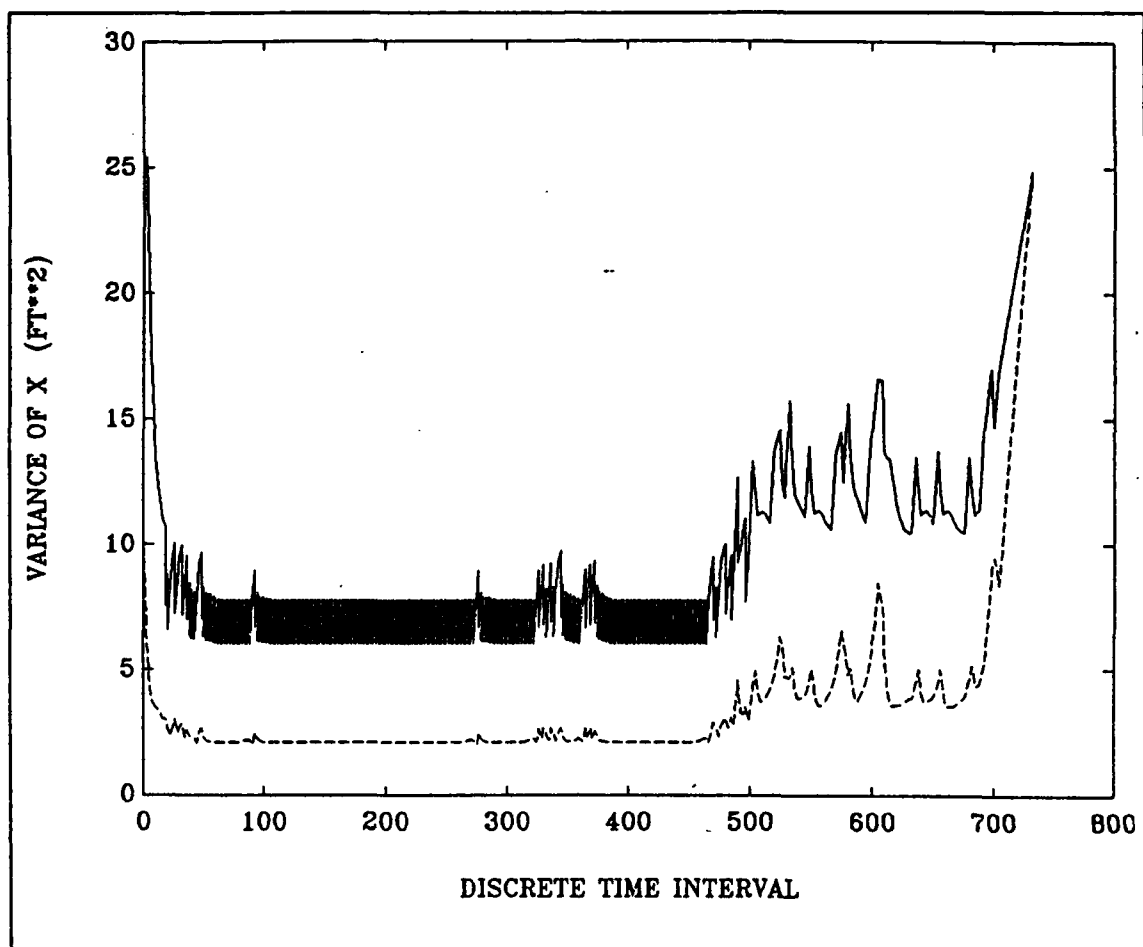


Figure 19. Kalman Filtered & Smoothed Variances Of X (Arrays 1 & 11).

smoothing algorithm's variance of the x estimate is, as expected, a significant improvement over the filter's performance alone. Note that the overlap region now settles to a single steady state variance of about 2.5 square feet, and that, although the variance jumps up over the latter portion of the data set as before, the values are again much smaller than the filtered result. Figure 20 on page 28 displays the Kalman filtered track of the data shown in Figure 5 for the X vs. Y coordinates case. The filtered track overshoots the raw track initially, indicating that better tuning may be required as discussed earlier, but then settles to an average of the two distinct array tracks throughout the overlap region. Finally, the filtered track closely follows the track after the overlap region has ended while effectively removing much of the random noise present.

Figure 21 on page 29 is the smoothed version of Figure 20 on page 28, and proves to

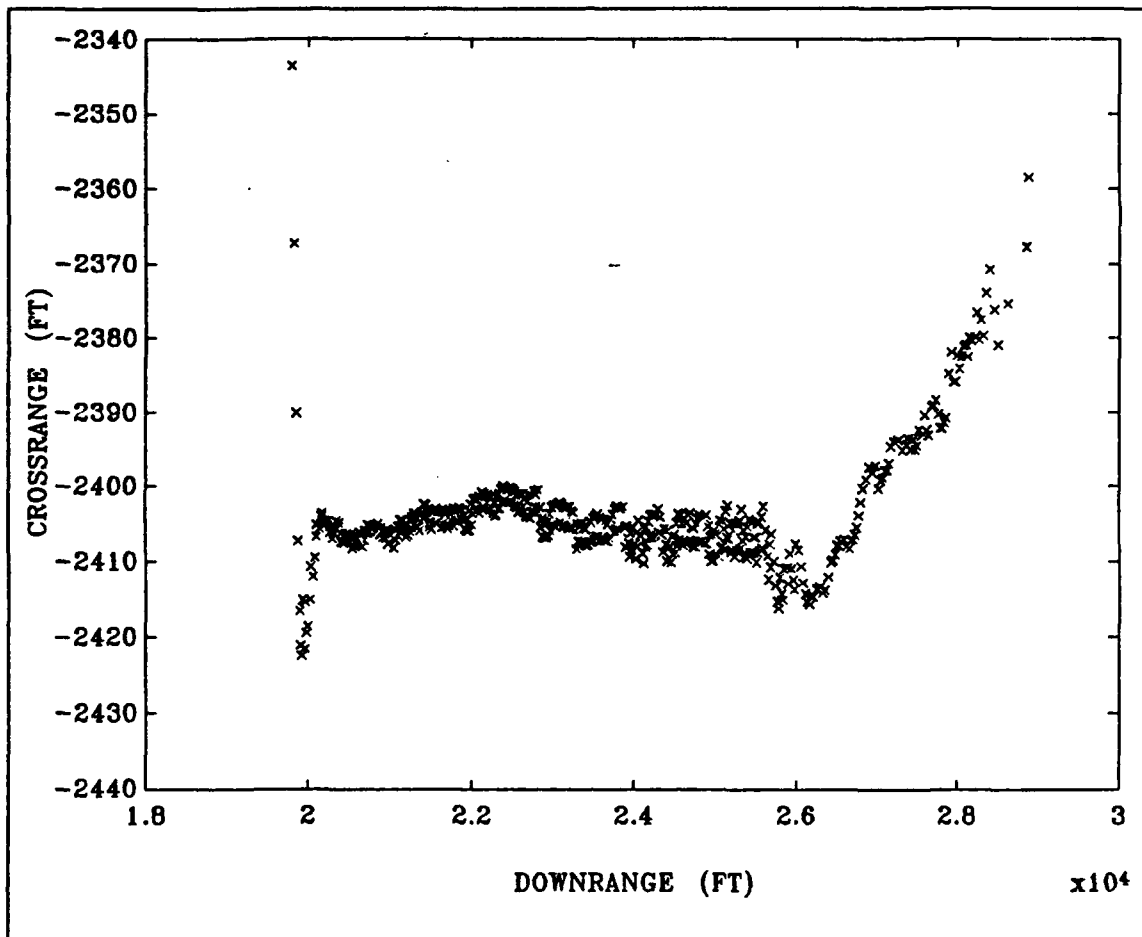


Figure 20. Kalman Filtered Track - X vs. Y (Arrays 1 & 11).

be an easy-to-see track which does not overshoot the target's maneuver. Thus, the smoothing routine has not only smoothed the filtered track considerably, but it has also compensated for the possible tuning problem. Figure 22 on page 30 is the smoothed track superimposed on the raw data, and it shows this clearly. Figure 23 on page 31 and Figure 24 on page 32 show the filtered and smoothed tracks respectively for the X vs. Z case. These plots exhibit the same characteristics as the X vs. Y plots discussed above as expected. Figure 25 on page 33 highlights the overall improvement gained by smoothing the raw data.

Another set of data which contained an overlap region was processed by the program. This data set was considerably larger than the previous case but was still within the capacity of the personal computer. This data is illustrated in Figure 26 on page 34

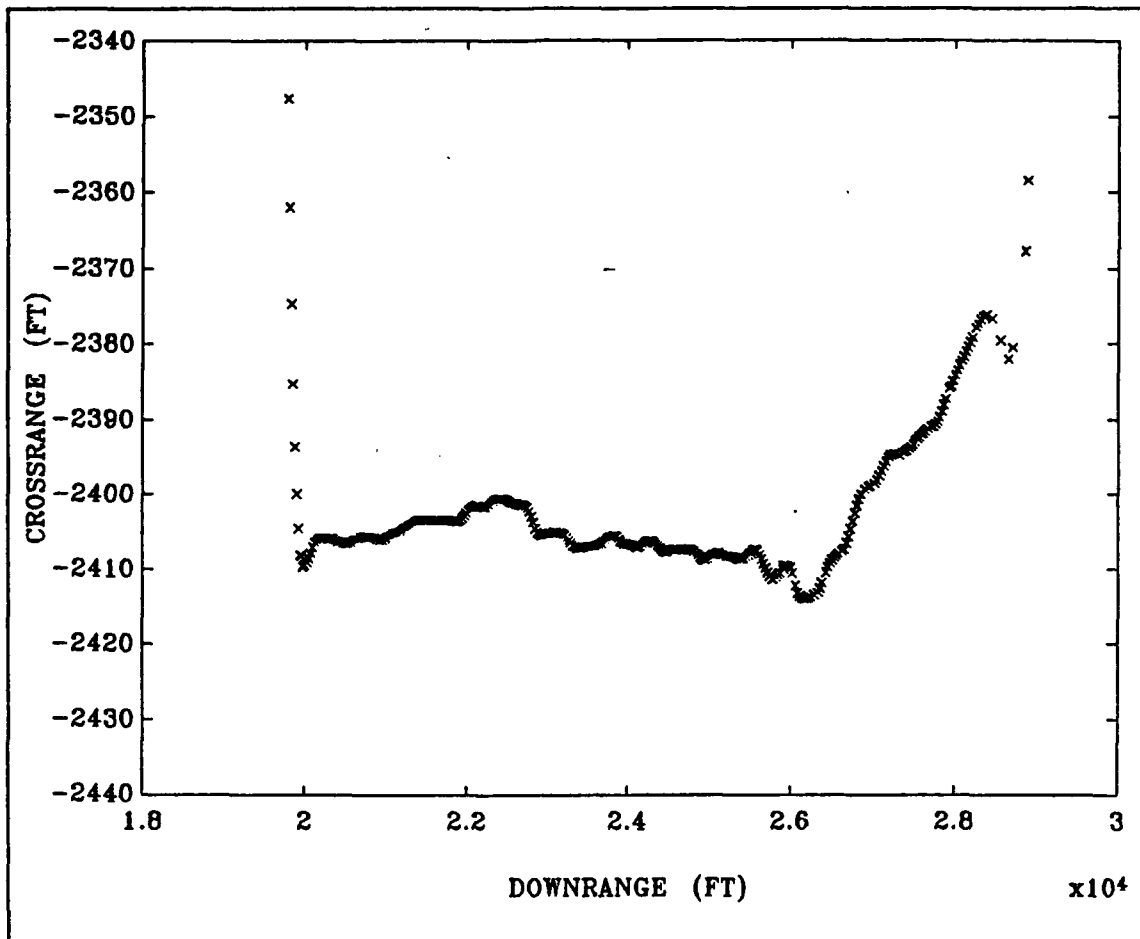


Figure 21. Smoothed Track - X vs. Y (Arrays 1 & 11).

and Figure 27 on page 35. Figure 28 on page 36 displays the filtered and smoothed values of the x estimate variances vs. time. As in the previous case, the missed samples and overlap region are apparent from the filtered results, and the smoothed results show much improvement. Figure 29 on page 37 shows the filtered results while Figure 30 on page 38 shows the smoothed results of the X vs. Y tracks for the arrays 2 & 12 overlap region, and it is clearly illustrated that the track quality is again vastly improved. Figure 31 on page 39 is the smoothed X vs. Y track superimposed on the raw data demonstrating that the smoothed data does indeed follow the raw data closely. Figure 32 on page 40 and Figure 33 on page 41 show the filtered and smoothed X vs. Z plots respectively with equally good results. Finally, Figure 34 on page 42 is also included to highlight the final results of the program. Notice that there is no overshoot

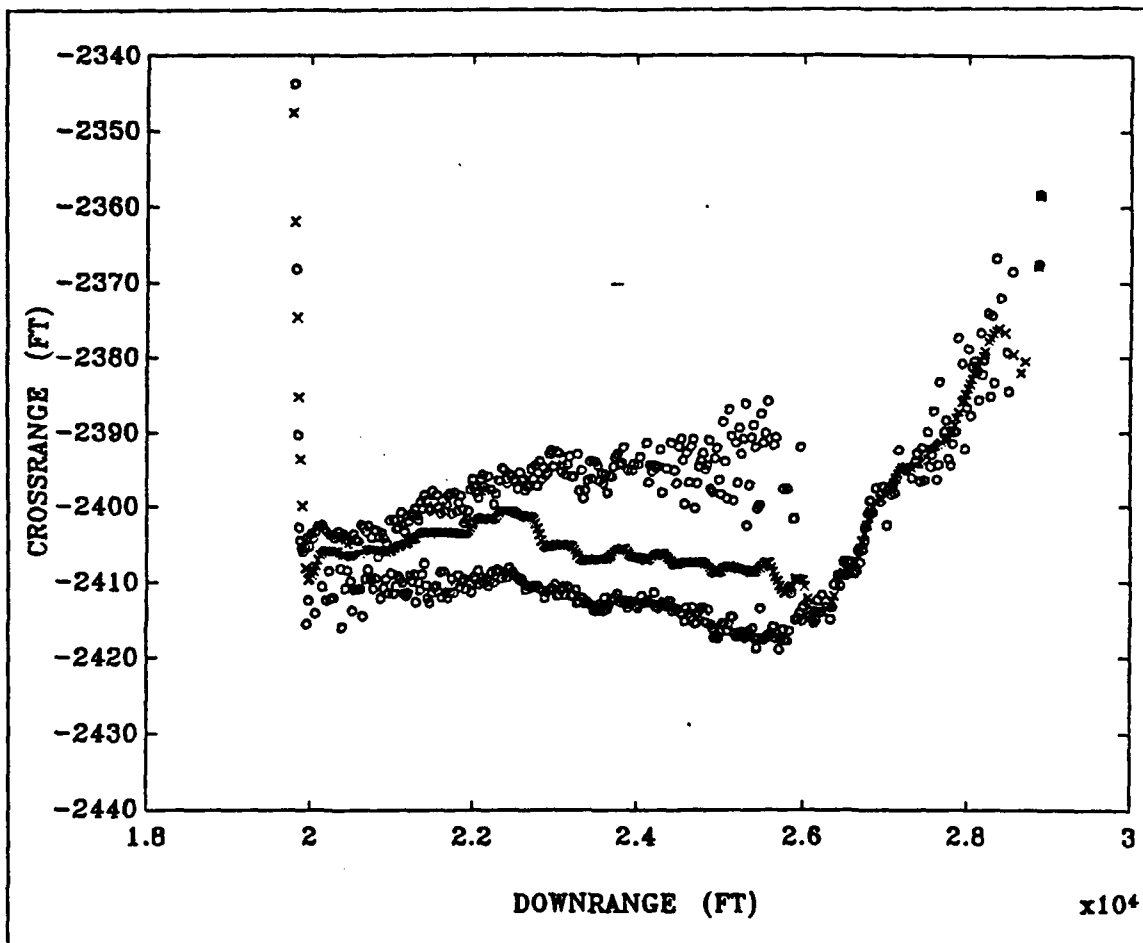


Figure 22. Raw & Smoothed Tracks - X vs. Y (Arrays 1 & 11).

in these filtered tracks as was seen in the previous case, indicating that the Kalman Filter was properly tuned for this run.

These results clearly demonstrate that the Kalman Filter with the Fixed Interval Smoothing Algorithm will essentially "solve" the array handoff problem by removing random noise and taking on a single average value through overlap regions based on information obtained from all hydrophones in question. The program effectively handles missed points and samples with multiple data especially where the smoothed output is considered. Therefore, with sufficient computer memory available, this program could accept a target's entire range track file filled with missed samples and overlapping data regions and produce the optimum track for the known information. This

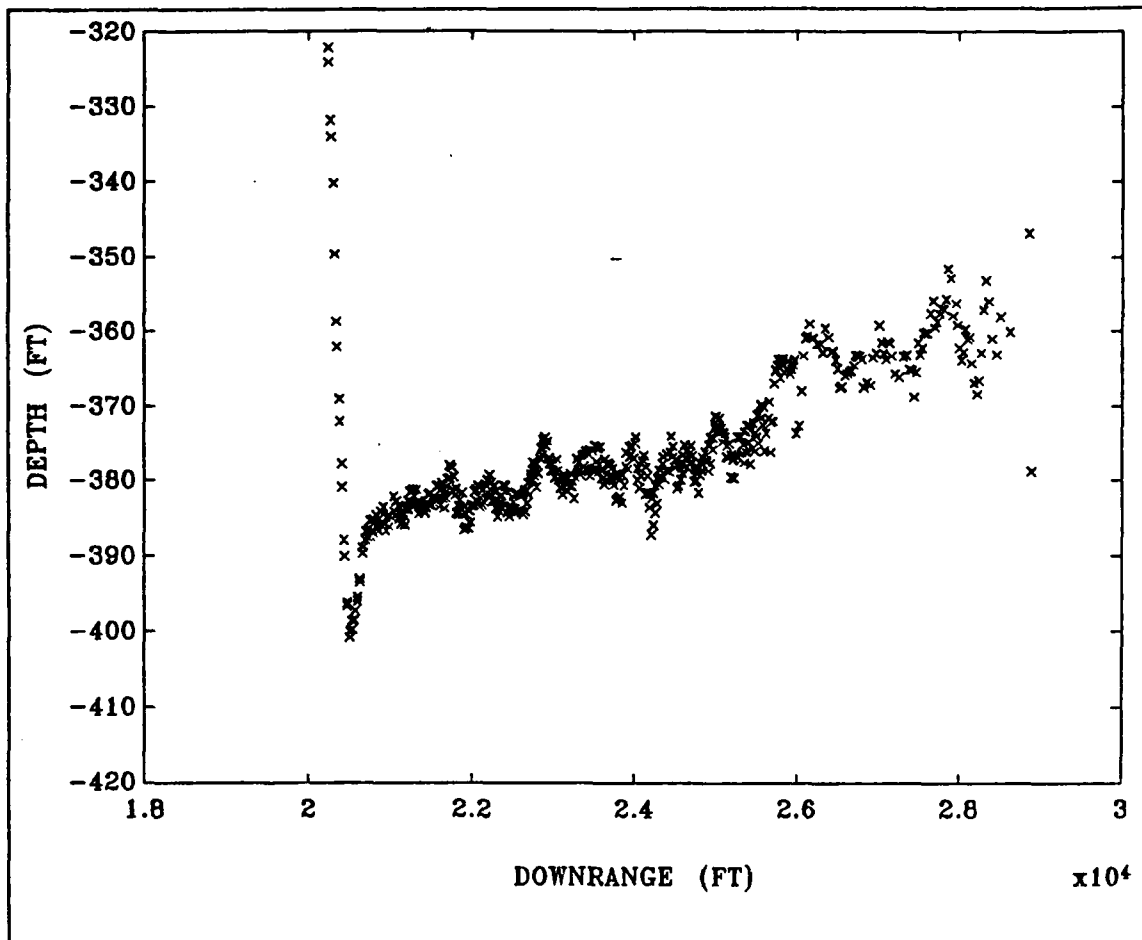


Figure 23. Kalman Filtered Track - X vs. Z (Arrays 1 & 11).

fact makes this an extremely useful post-data analysis tool for evaluating typical range generated tracks.

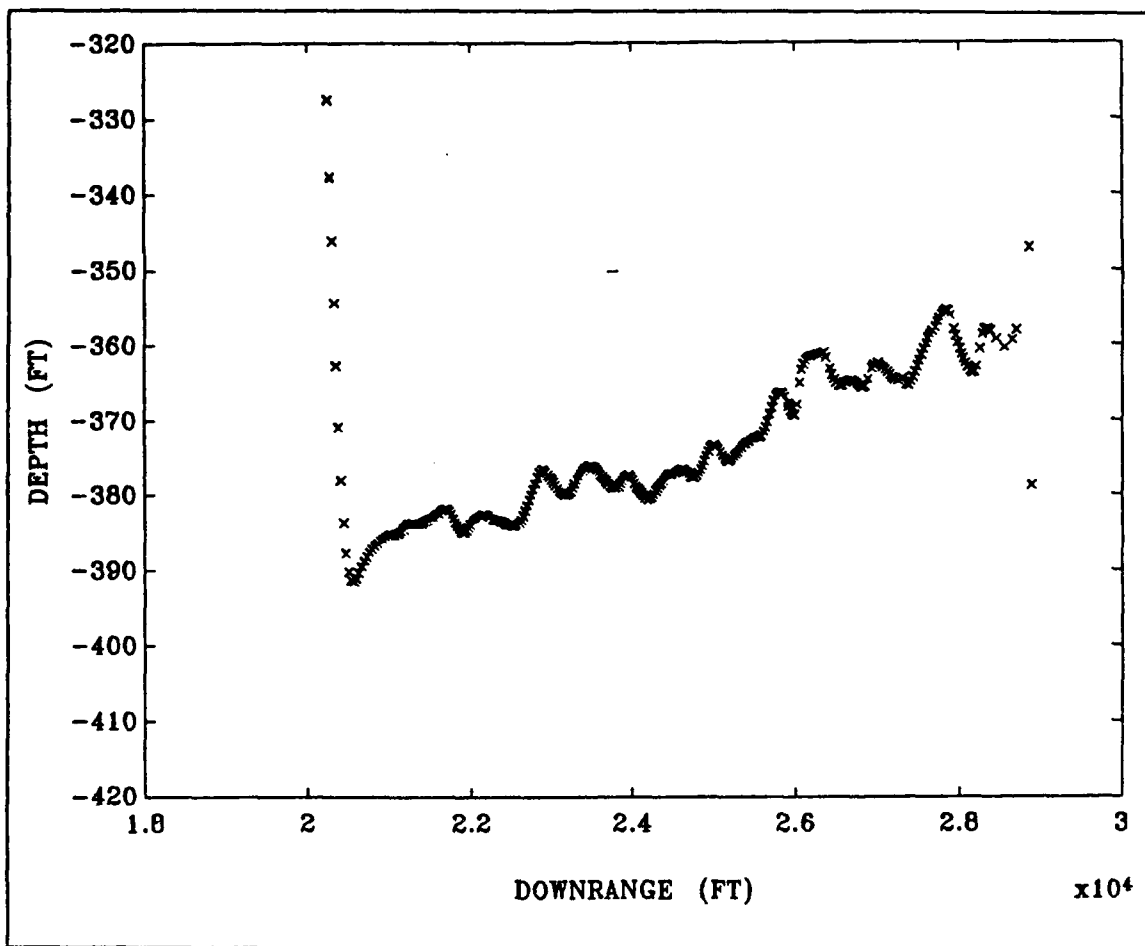


Figure 24. Smoothed Track - X vs. Z (Arrays 1 & 11)

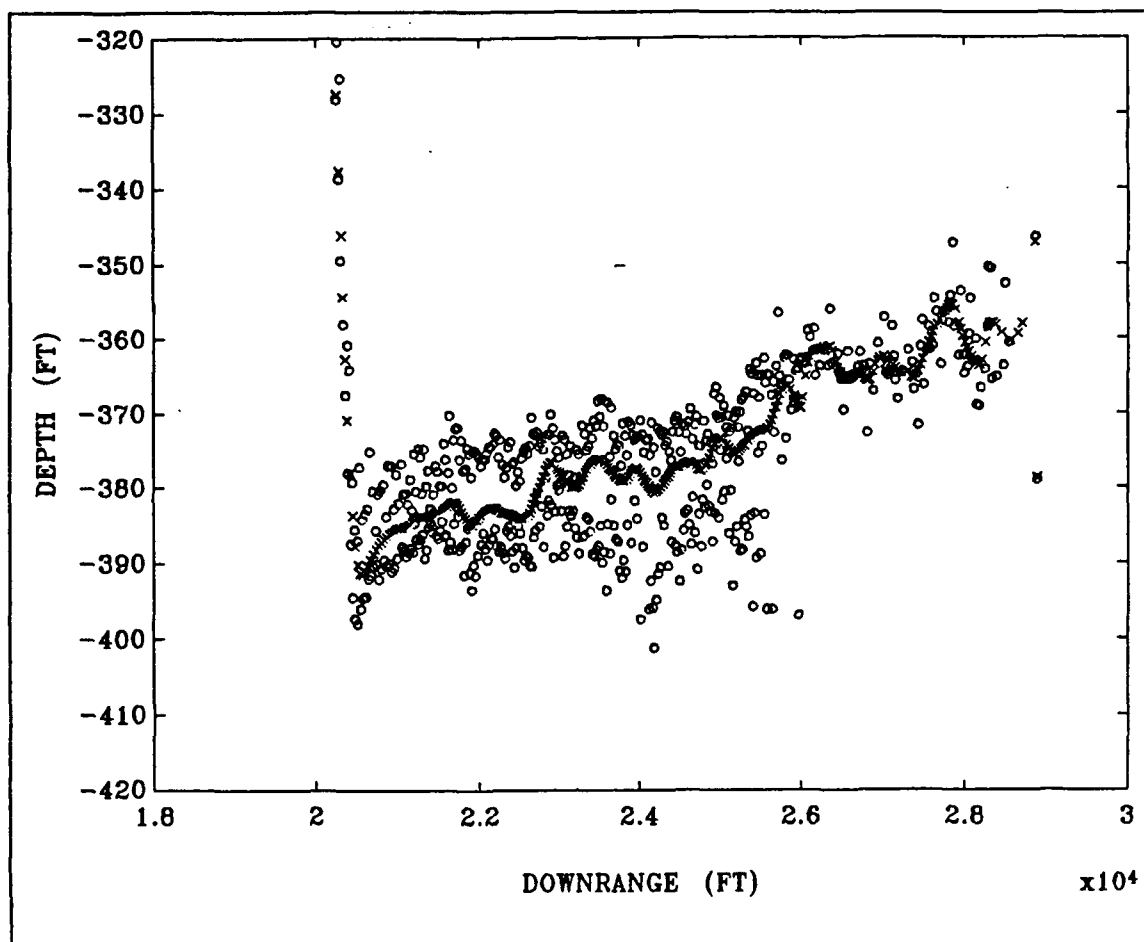


Figure 25. Raw & Smoothed Tracks - X vs. Z (Arrays 1 & 11).

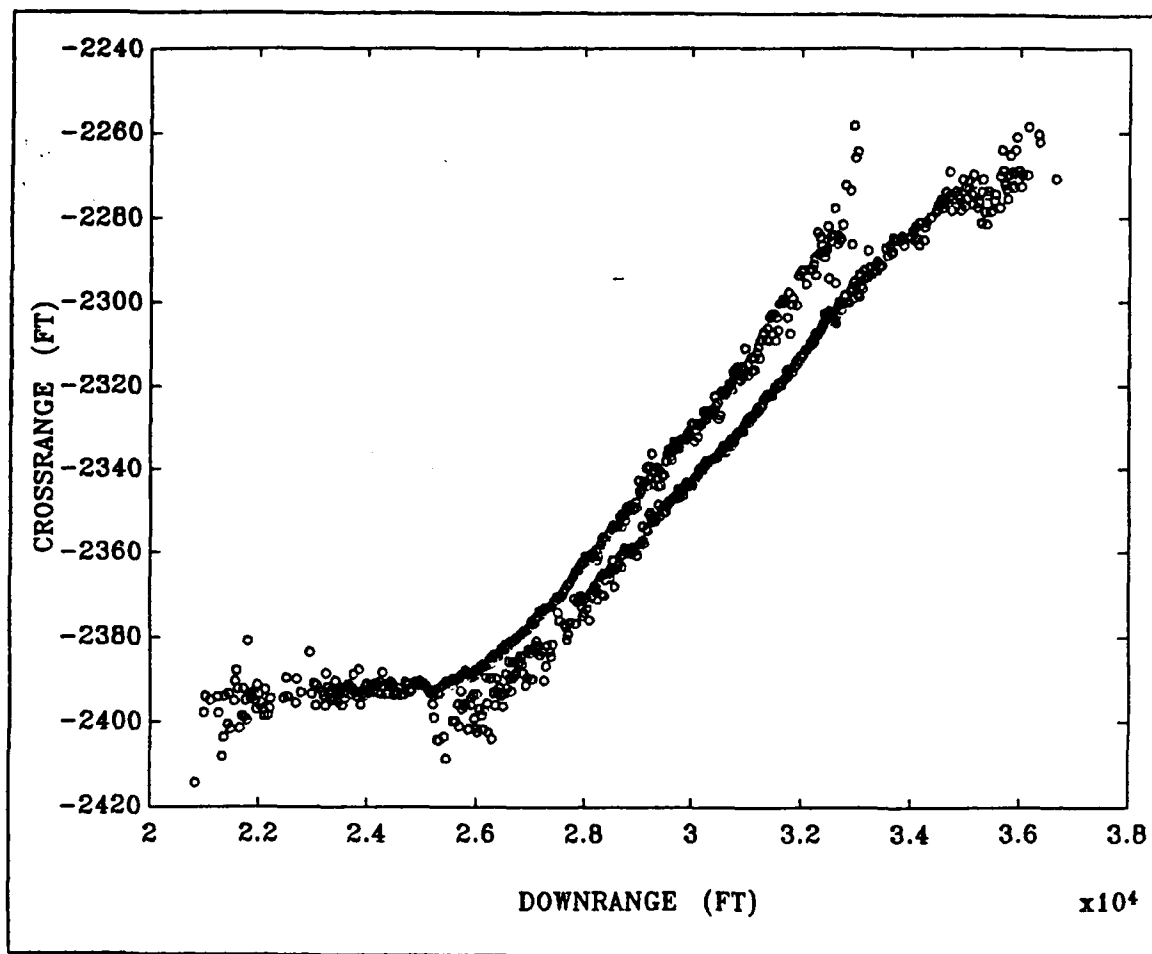


Figure 26. Raw Track - X vs. Y (Arrays 2 & 12).

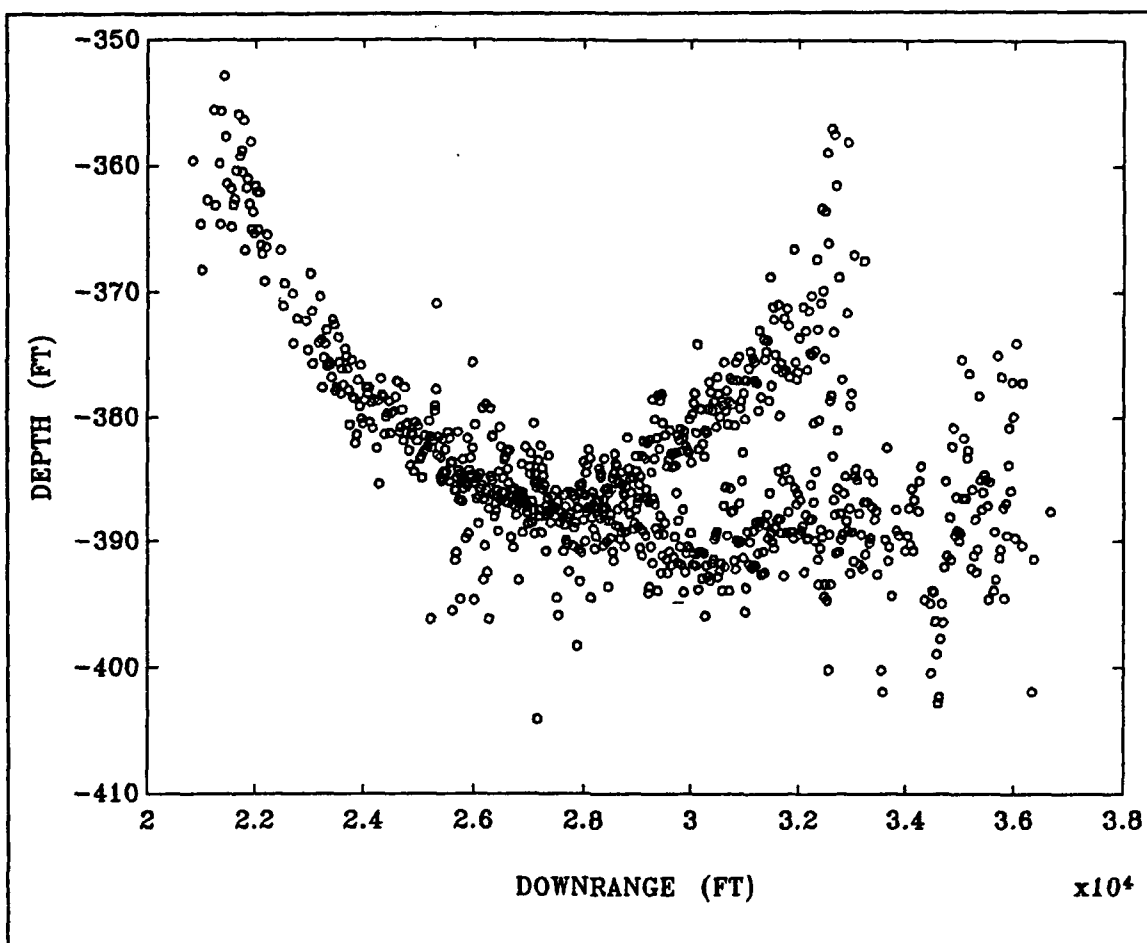


Figure 27. Raw Track - X vs. Z (Arrays 2 & 12).

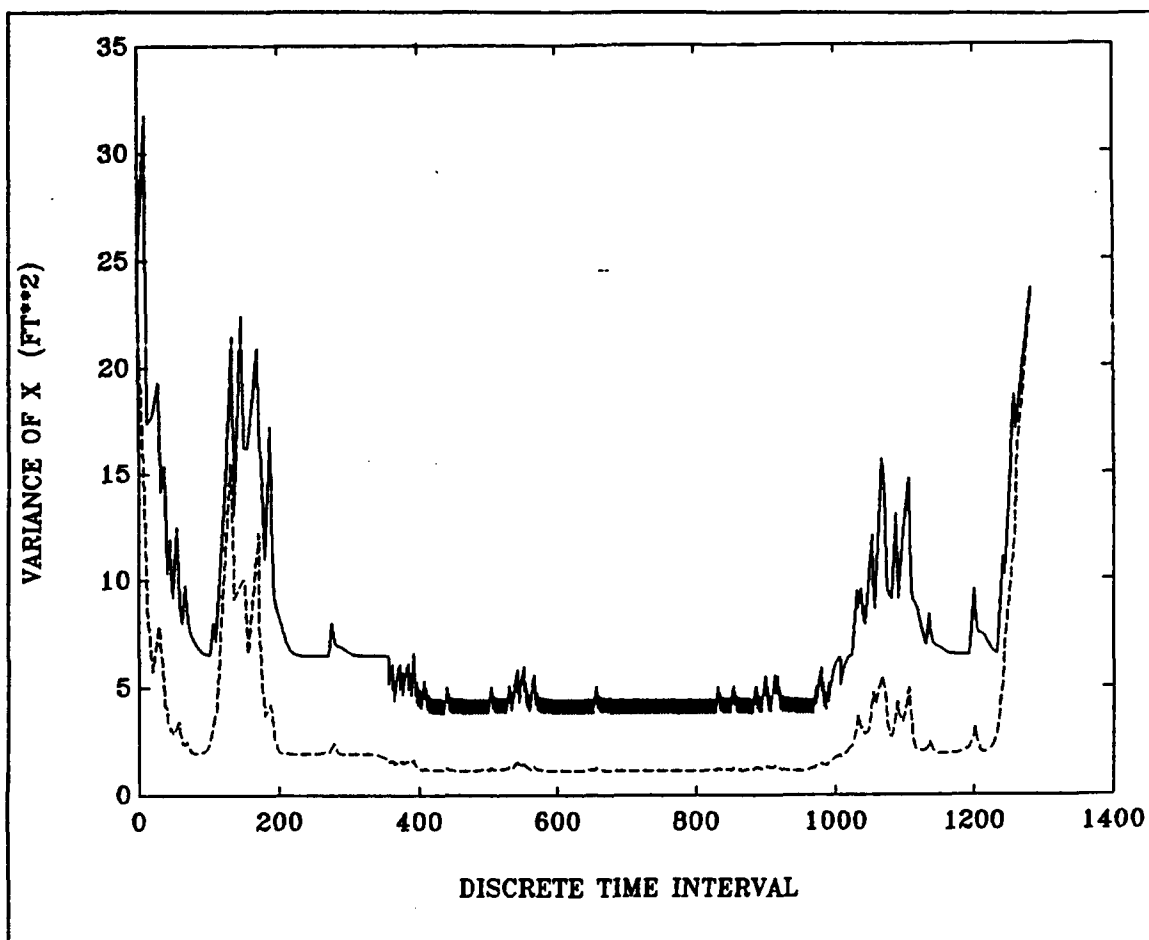


Figure 28. Kalman Filtered & Smoothed Variances Of X (Arrays 2 & 12).

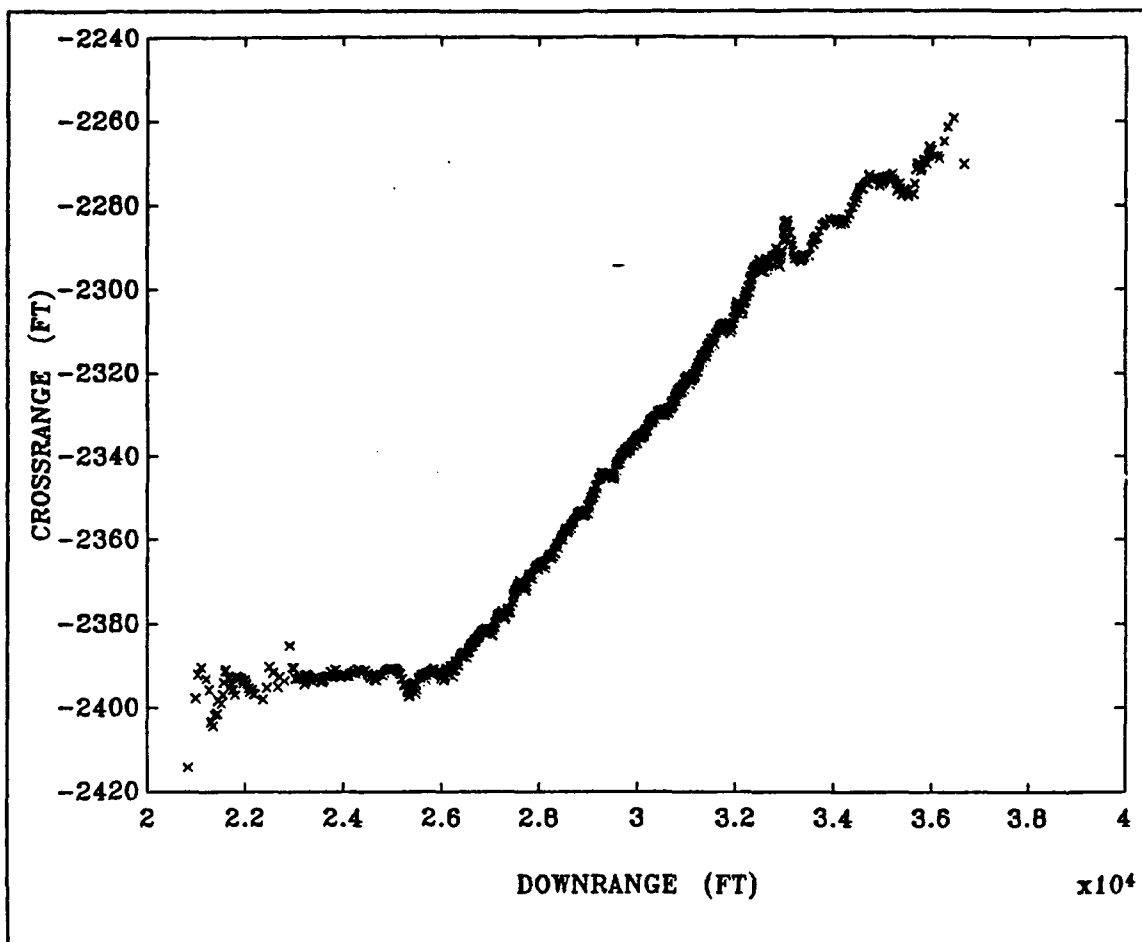


Figure 29. Kalman Filtered Track - X vs. Y (Arrays 2 & 12).

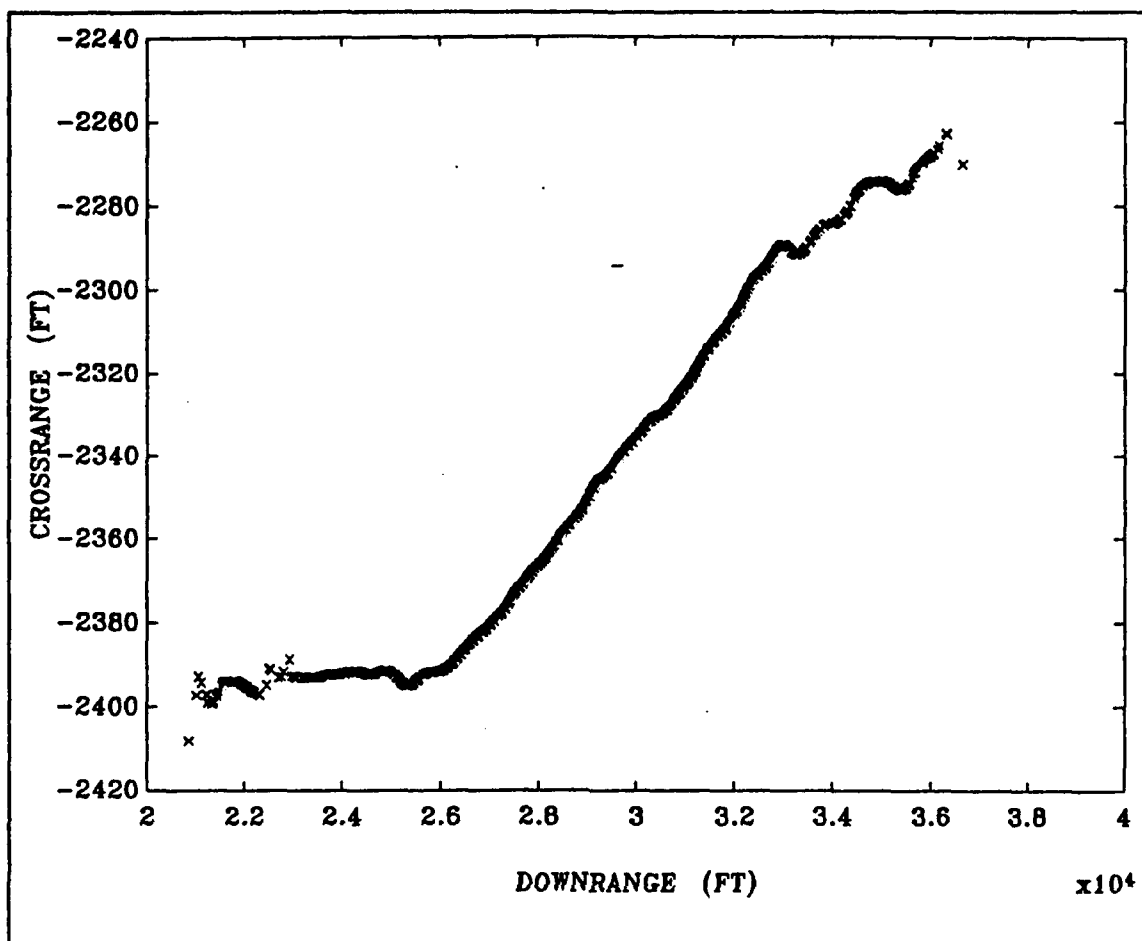


Figure 30. Smoothed Track - X vs. Y (Arrays 2 & 12).

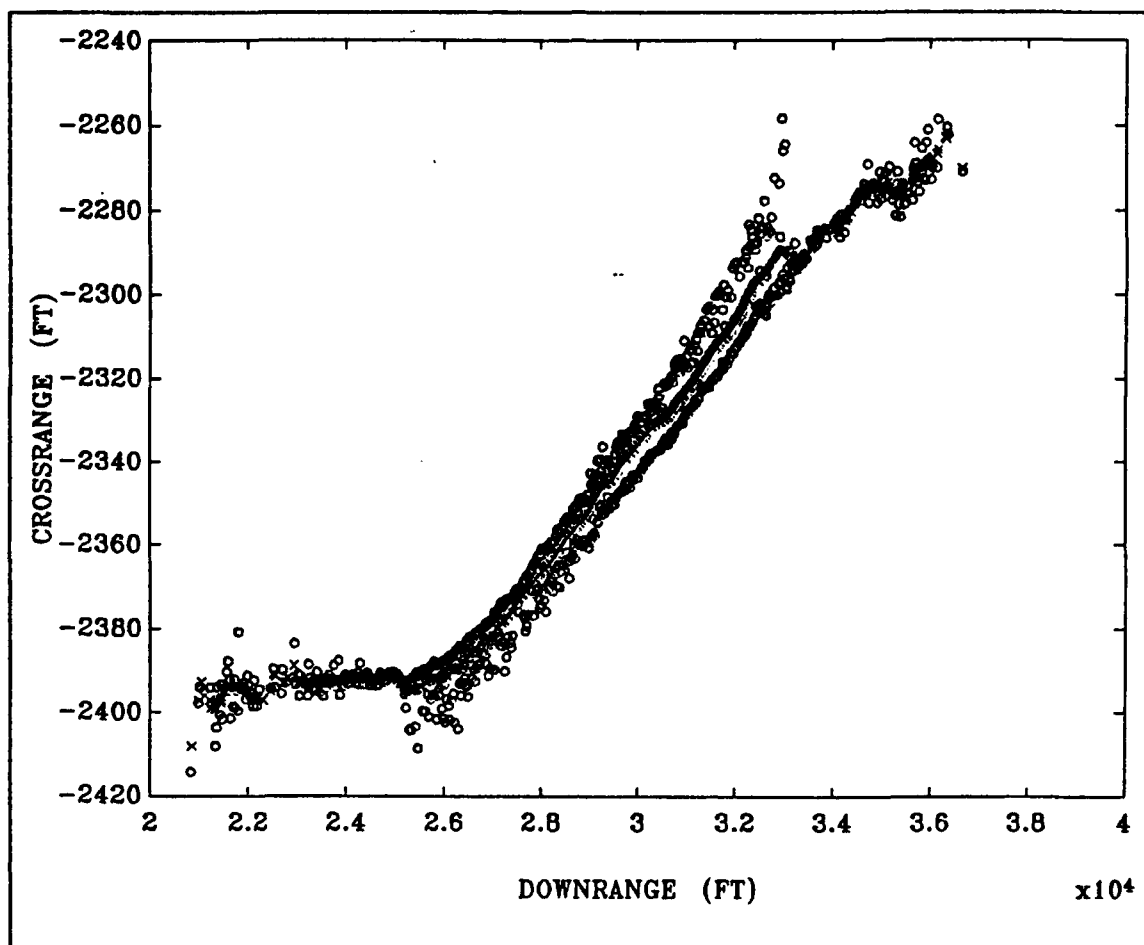


Figure 31. Raw & Smoothed Tracks - X vs. Y (Arrays 2 & 12).

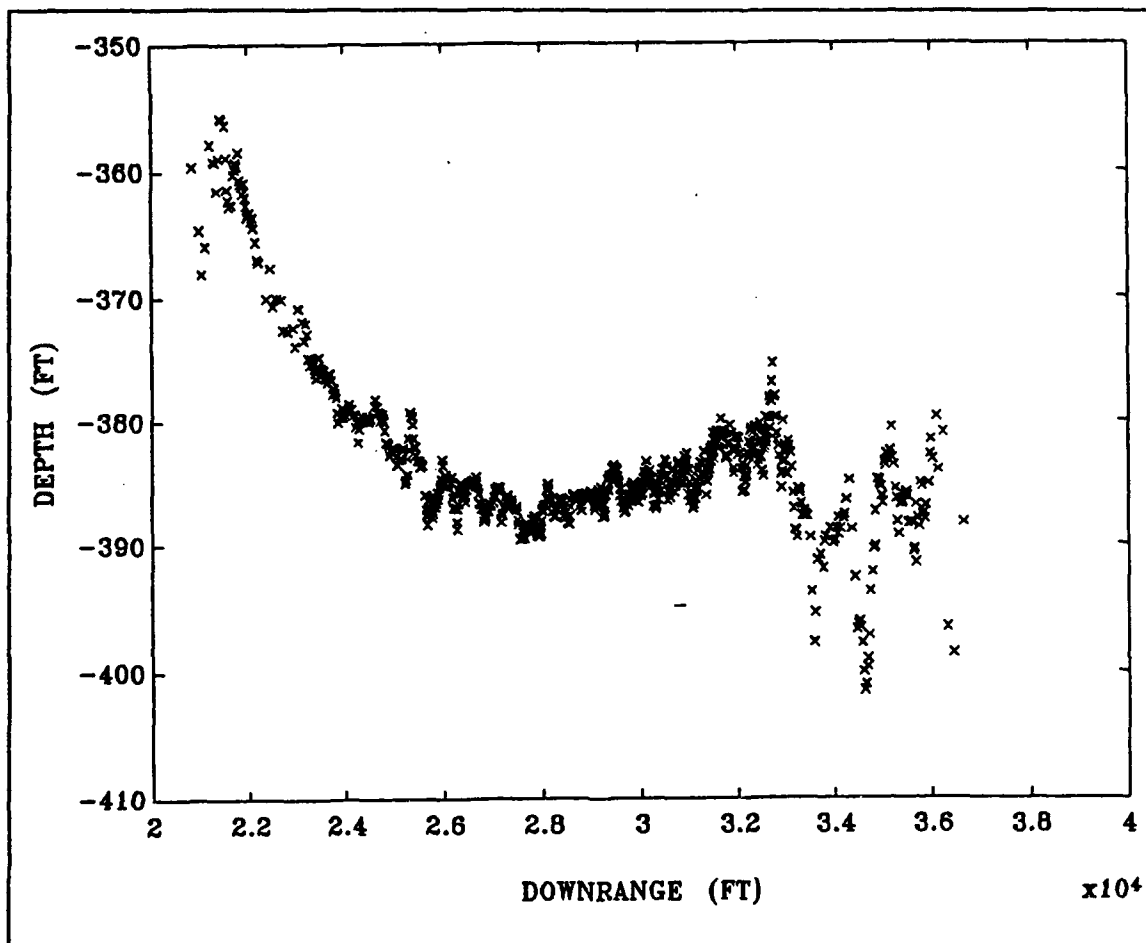


Figure 32. Kalman Filtered Track - X vs. Z (Arrays 2 & 12).

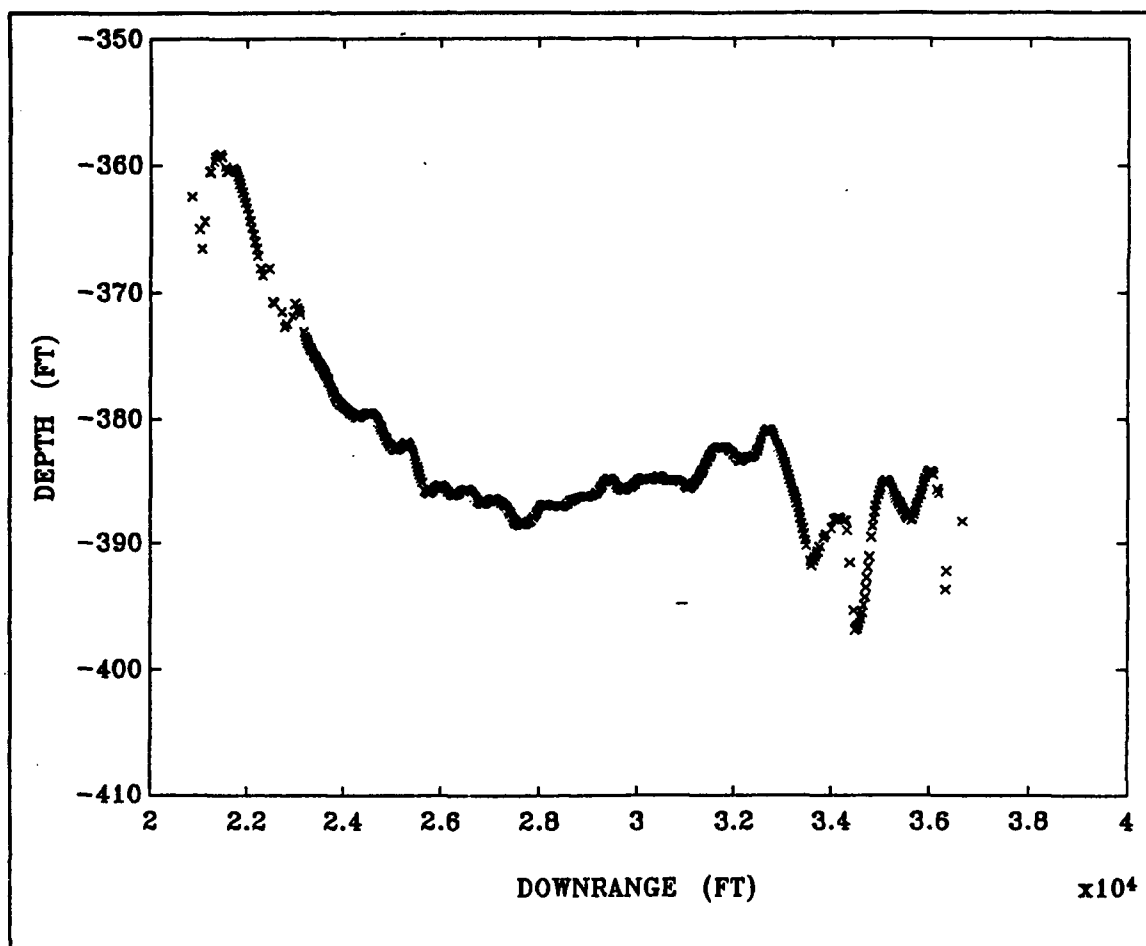


Figure 33. Smoothed Track - X vs. Z (Arrays 2 & 12).

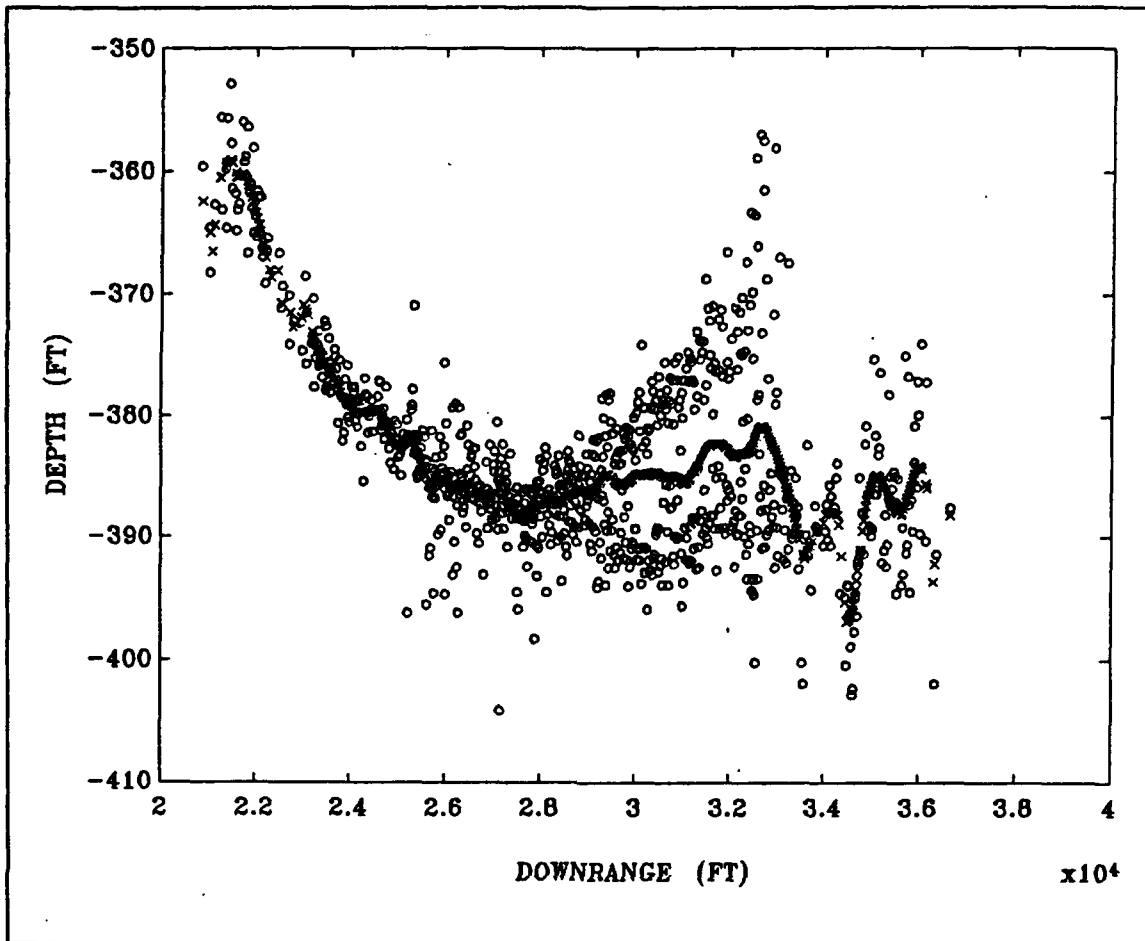


Figure 34. Raw & Smoothed Tracks - X vs. Z (Arrays 2 & 12).

V. CONCLUSION

The results above have demonstrated the effectiveness of this Kalman Filter Fixed Interval Smoothing Algorithm in smoothing a target track for use in post-data analysis. A raw target track characterized by discontinuities in array overlap regions and random noise throughout can easily be transformed into the optimal track given the known information. All the user need do is set up an appropriate input file consisting of sample count and X, Y, and Z coordinates. The program listed in Appendix B is designed to take current data files from the Keyport Torpedo Station and reformat them into suitable input files for the main smoothing program using data from any two arrays. This program could easily be modified to include data from as many arrays as desired. Thus, the original goal of this research, to provide a smoothing algorithm tested on actual range data which effectively deals with the array handoff problem, has been successfully achieved. In addition, this algorithm's success suggests some other interesting possibilities for improving short base-line array tracking range capabilities.

Another possible use for this algorithm is range calibration. As has been discussed, the current tracks generated by the short base-line arrays contain random errors and bias errors due to a myriad of sources. If a test craft could be tracked accurately independent of the acoustic range, it could be used to generate a true track for comparison to the smoothed track produced by this program. It follows that since the Kalman Filter Smoothing routine effectively reduces random errors while having no effect on bias errors, any bias errors present in the smoothed track would be highlighted; whereas, they may have been lost before in the random noise. These bias errors could then be accounted for in the software that calculates the target's position. Calibration runs could be made as frequently as desired, and in this way, bias errors could be promptly and easily compensated for.

The Kalman Filter portion of the program presented here is designed to operate on a file of existing data points. However, it would be a relatively simple matter to alter this program to read one data point, operate on the data, generate a filtered data point, and store the required information for use in the smoothing portion of the program. The smoothing routine in this program is desired to run after all the data has been Kalman filtered, so, no changes are required in this portion of the program as long as the last data point of the track is properly flagged. The implications of these facts is that this

program could be used to improve the real time tracks generated by the range as well as providing a smoothed track after the run is complete for post-data analysis and range calibration. The program need only be given the computed output of the tracking computer and then allowed to supply its filtered result to the algorithm which generates the track displays. Furthermore, improved tracking accuracy could be obtained using an adaptive excitation matrix scheme. It was shown earlier how $Q(k)$ can affect the filtered results. Assuming frequency information could be provided to the program in real time, the magnitude of $Q(k)$ could be altered in response to doppler shifts in received frequency. In other words, if a target maneuver is indicated by a doppler shift, the magnitude of $Q(k)$ is increased to allow the filter to track through the maneuver. Once the doppler information indicates that the target is no longer maneuvering, $Q(k)$ is decreased to smooth out the filtered track. In this way, the current real time tracking capabilities of these ranges could be improved with a reasonable amount of changes in the tracking software required.

In conclusion, the approach used here of Kalman Filtering the target's computed position in X, Y, and Z coordinates is considered to be successful. This method is an effective compromise between the theory of optimal linear estimation and the operationally oriented user who may be leery of trusting massaged data, because the raw data is not lost. Therefore, the user can always make comparisons between raw and treated data. In addition, this program is seen as flexible enough to be applied to a number of possible applications concerning the existing short base-line array tracking ranges. Finally, it is an excellent post-data analysis tool in its present form.

APPENDIX A. KALMAN FILTER FIXED INTERVAL SMOOTHING ALGORITHM

A. MAIN PROGRAM

A brief description of the main program and subroutines is given. Initially, the program reads in data from a user-specified data file and sets a point counter represented by the variable IR1. After initialization of the necessary variables, the program determines the time slot to the next sequential data point in the file and converts it to seconds for use by the subroutine which calculates the state transition matrix ϕ . Next, the excitation matrix $Q(k)$ is computed. Notice that this calculation requires the time elapsed from the last sample. The gain matrix $G(k)$ is then computed followed by the filtered estimates of the states. The program then stores the filtered estimates along with the information needed for the smoothing algorithm, and then initializes the array for later storage of the smoothed estimates. The variance of the x estimate is sent to a subroutine to be placed in a data file.

The Fixed Interval Smoothing portion of the program first sets up its loop counter to count back from the filtered loop endpoint. Next, ϕ is recomputed using times between samples that are consistent with the Kalman Filter case. Pertinent information stored in the Kalman Filter loop is recovered, and the smoothed estimates are then solved for. The smoothed x estimate variance is then sent to a subroutine which includes it in an output file, and finally, the filtered and smoothed estimates are written to output data files.

B. SUBROUTINES

- **PLOTPKKS**
outputs the variance of the x estimate vs. discrete time interval for the Kalman Filtered case and the Smoothed case
- **PHIDEL**
computes the state transition matrix based on time between samples using a linear second order model
- **GAIN**
computes $P(k/k-1)$, $P(k/k)$, and the gain matrix $G(k)$
- **ADD**
adds two input matrices
- **SUB**
subtracts second input matrix from first

- **PROD**
multiplies two input matrices
- **TRANS**
calculates the transpose of the matrix input
- **RECIP**
algorithm taken from [Ref. 17: p. 163] which inverts the input matrix
- **INITS**
initializes matrices at the beginning of the main program
- **CHANGE**
allows user to change the magnitude of the excitation matrix $Q(k)$ and the initial value of the estimate's covariance of error matrix $P(k/k)$

\$LARGE

```
C---KALMAN FILTER---KALMAN SMOOTHING ALGORITHM-----
C---THIS PROGRAM IS DESIGNED TO PROCESS 3/D DATA FILES FROM
C---THE UNDERSEA TRACKING RANGE AT KEYPORT, WA.  A KALMAN FILTER
C---IS APPLIED TO THE TRACK DATA WHICH CONSISTS OF X,Y, AND Z
C---COORDINATES.  THEN, A KALMAN FILTER SMOOTHING ROUTINE GENERATES
C---SMOOTHED POINTS IN X,Y, AND Z.  THE PROGRAM GENERATES OUTPUT
C---FILES WHICH CONTAIN THE VARIANCE OF THE X ESTIMATE VS DISCRETE TIME
C---FOR BOTH THE FORWARD KALMAN FILTER CASE AND THE KALMAN SMOOTHED
C---CASE.  FILES ARE ALSO GENERATED WHICH CONTAIN THE FILTERED
C---X,Y,AND Z ESTIMATES AND THE SMOOTHED X,Y,AND Z ESTIMATES.
C---THIS PROGRAM IS DESIGNED TO RUN ON THE IBM/AT PERSONAL
C---COMPUTER BUT DUE TO THE SIZE OF THE DATA SETS INVOLVED, PLOTTING
C---CANNOT BE DONE WITH THIS PROGRAM.  PLOTTING OF OUTPUT DATA IS
C---DONE USING MATLAB.  THE PROGRAM GIVES THE USER THE OPTION OF
C---CHANGING THE VALUE OF THE INITIAL COVARIANCE MATRIX AND THE
C---EXCITATION PROCESS VECTOR.  THE USER IS ALSO REQUIRED TO PROVIDE
C---THE NAMES OF THE INPUT AND OUTPUT DATA FILES.
```

C

C

C

***** DECLARATION OF VARIABLES *****

```
COMMON  W(6,6),XKK(6),PHI(6,6),Q(6,6),XKKM1(6),
* A(6,6),B(6,6),H(6,6),HI(6,6),PKKM1(6,6),PKK(6,6)
COMMON /CBLK/ RAW(3,1000),FILT(3,1000),SMOOTH(3,1000),PTC(1000),
* XYRANGE(3,2),PKKONEONE(1000),IR1
CHARACTER INFIL*13, FILTER*13, OUTFIL*13
DIMENSION R(6,6),G(6,6),XRR(1000),YRR(1000),ZRR(1000),PHIT(6,6),
*ZZ(6),E(6),GE(6),PKKS3D(6,6,1000),PKKM1S(6,6,1000),XKKS(6,1000),
*C(2,2),D(2,2),PKKS(6,6),P2(6,6),PNNM1(6,6),ZKKM1(6),AKT(6,6),
*SS3(6,6),SS3R(6,6),AK(6,6),XNNM1(6),SS2(6),XP1(6),TEMP4(6,6),
*TEMP5(6,6),TEMP6(6,6),TEMP3(6,6),TEMP2(6),TEMP1(6,6),TMP(6)
```

C

C

C

***** INPUT DATA, DESIGNATE FILENAMES *****

```
WRITE(*,*) 'ENTER NAME OF INPUT DATA FILE'
READ(*, '(A)') INFIL
WRITE(*,*) 'ENTER NAME OF FILTERED DATA FILE'
READ(*, '(A)') FILTER
WRITE(*,*) 'ENTER NAME OF SMOOTHED DATA FILE'
READ(*, '(A)') OUTFIL
```

```

      OPEN(1,FILE= INFIL,STATUS= 'OLD')
      OPEN(2,FILE= FILTER,STATUS= 'NEW')
      OPEN(3,FILE= OUTFIL,STATUS= 'NEW')
      IR1=1
15  READ(1,*,END=20)PTC(IR1),XRR(IR1),YRR(IR1),ZRR(IR1)
      IR1=IR1+1
      GOTO 15
20  CONTINUE
      IR1 = IR1 - 1

C
C      **** INITIALIZE VARIABLES AND DATA ARRAYS ****
C
      L = 1
      M = 3
      N = 6
      MC = 6
      ND = 6
      LD = 6
      DT = 0.
      K = 0
      W(1,1) = .0001
      R(1,1) = 25
      R(2,2) = 25
      R(3,3) = 25
      R(4,4) = 25
      R(5,5) = 25
      R(6,6) = 25
      CALL INITS

C
C      **** CHECK INITIAL PARAMETERS ****
C
      CALL CHANGE(W,PKKM1)

C
C      *****
C      KALMAN FILTER ROUTINE
C      *****

100 K = K + 1

C
C      **** GENERATE PHI AND Q MATRICES ****
C
      IF (K .EQ. IR1) THEN
        DT = 2.666
      ELSE
        DT = 1.333*(PTC(K+1) - PTC(K))
      END IF
      CALL PHIDEL(DT,N,A,PHI)
      IF (K .EQ. 1) THEN
        DT = 2.666
      ELSE
        DT = 1.333*(PTC(K) - PTC(K-1))
      END IF
      Q(1,1) = (DT**4/4) * W(1,1)
      Q(2,2) = DT**2 * W(1,1)
      Q(3,3) = Q(1,1)
      Q(4,4) = Q(2,2)

```

```

      Q(5,5) = Q(1,1)
      Q(6,6) = Q(2,2)
      RAW(1,K) = XRR(K)
      RAW(2,K) = YRR(K)
      RAW(3,K) = ZRR(K)
C
C      **** CALCULATE GAIN MATRIX AND SOLVE FOR      ****
C      **** X(K/K) = X(K/K-1) + G(K)*[ Z(K) - Z(K/K-1)] ****
C      **** WHERE Z(K) IS THE OBSERVABLE      ****
C
      CALL GAIN(PKK,PKKM1,Q,R,PHI,H,N,G,HI,ND,MD,LD)
      CALL PROD(PHI,XKK,N,N,1,XKKM1,ND,MD,LD)
      CALL PROD(H,XKKM1,N,N,1,ZKKM1,ND,MD,MD)
      ZZ(1)=XRR(K)
      ZZ(2)=YRR(K)
      ZZ(3)=ZRR(K)
      CALL SUB(ZZ,ZKKM1,M,1,E,ND,MD)
      CALL PROD(G,E,N,M,1,GE,ND,MD,LD)
      CALL ADD(XKKM1,GE,N,1,XKK,ND,MD)
C
C      **** STORE X(K/K),P(K/K),P(K/K-1),INITIALIZE THE SMOOTHED ****
C      **** DATA ARRAY, AND STORE THE FILTERED ESTIMATES. ****
C
      DO 40 I = 1,6
        DO 40 J = 1,6
          PKKS3D(I,J,K) = PKK(I,J)
40      PKKM1S(I,J,K) = PKKM1(I,J)
        DO 50 I=1,6
          XKKS(I,K) = XKK(I)
        DO 60 I = 1,3
          FILT(I,K) = XKK(2*I-1)
60      SMOOTH(I,K) = FILT(I,K)
      PKKONEONE(K) = PKKS3D(1,1,K)
      IF (K.LT.IR1) GOTO 100
      CT = 1
      CALL PLOTPKKS(CT)
C
C      ****
C      KALMAN SMOOTHING ROUTINE
C      ****
C
      DO 600 K=1,IR1 - 1
        KI= IR1 - K
C
C      **** GENERATE PHI MATRIX AND RETRIEVE STORED X(K/K), ****
C      **** P(K,K), AND P(K+1/K) ****
C
      DT = 1.333*(PTC(KI+1) - PTC(KI))
      CALL PHIDEL(DT,N,A,PHI)
      DO 501 I = 1,6
        XP1(I) = XKKS(I,KI)
501 CONTINUE
      DO 502 I = 1,6
        DO 502 J = 1,6
          P2(I,J) = PKKS3D(I,J,KI)
          SS3(I,J) = PKKM1S(I,J,KI)

```

```

502 CONTINUE
C
C      ****          SOLVE FOR SMOOTHED ESTIMATE          ****
C      ****      X(K/N) = X(K/K) + A(K)*[X(K+1/N) - X(K+1/K)]      ****
C
      CALL TRANS(PHI,N,N,PHIT,ND,MD)
      CALL RECIP(SS3,SS3R,N)
      CALL PROD(PHIT,SS3R,N,N,N,TEMP1,ND,MD,LD)
      CALL PROD(P2,TEMP1,N,N,N,AK,ND,MD,MD)
      DO 504 I = 1,6
        XNNM1(I) = XKKS(I,KI+1)
504 CONTINUE
      CALL PROD(PHI,XP1,N,N,L,SS2,ND,MD,L)
      CALL SUB(XNNM1,SS2,N,1,TEMP2,ND,L)
      CALL PROD(AK,TEMP2,N,N,1,TEMP3,ND,MD,L)
      CALL ADD(XP1,TEMP3,N,1,TMP,ND,MD)
      DO 505 I = 1,6
505      XKKS(I,KI) = TMP(I)
      DO 506 I = 1,6
        DO 506 J = 1,6
          PNNM1(I,J) = PKKS3D(I,J,KI+1)
506 CONTINUE
      CALL SUB(PNNM1,SS3,N,6,TEMP4,ND,MD)
      CALL TRANS(AK,N,N,AKT,ND,MD)
      CALL PROD(TEMP4,AKT,N,N,N,TEMP5,ND,MD,MD)
      CALL PROD(AK,TEMP5,N,N,6,TEMP6,ND,MD,MD)
      CALL ADD(P2,TEMP6,N,6,PKKS,ND,MD)
      DO 508 I = 1,6
        DO 508 J = 1,6
508      PKKS3D(I,J,KI) = PKKS(I,J)
C
C      ****      STORE X(K/K),P(K/K),AND OUTPUT SMOOTHED DATA      ****
C
      DO 520 I = 1,3
520      SMOOTH(I,KI) = XKKS(2*I-1,KI)
          PKKONEONE(KI) = PKKS3D(1,1,KI)
600 CONTINUE
      CT = 0
      CALL PLOTPKKS(CT)
      DO 650 K = 1,IR1
        WRITE(2,*)(FILT(I,K),I=1,3)
        WRITE(3,*)(SMOOTH(I,K),I=1,3)
650 CONTINUE
      END
C
C
C      *****
C      SUBROUTINE WHICH OUTPUTS THE VARIANCE OF THE X ESTIMATE VS
C      DISCRETE TIME FOR BOTH THE FILTERED CASE AND THE SMOOTHED CASE
C      *****
C
      SUBROUTINE PLOTPKKS(CT)
      COMMON /CBLK/ RAW(3,1000),FILT(3,1000),SMOOTH(3,1000),PTC(1000),
      * XYRANGE(3,2),PKKONEONE(1000),IR1
      DIMENSION TK(1000)
      CHARACTER FWDVAR3*13, BKVAR*13

```

```

      IF (CT .EQ. 1) THEN
        WRITE(*,*) 'ENTER NAME OF FILTERED VARIANCE FILE'
        READ(*, '(A)') FWDVAR
        OPEN(4, FILE= FWDVAR, STATUS= 'NEW')
      ELSE
        WRITE(*,*) 'ENTER NAME OF SMOOTHED VARIANCE FILE'
        READ(*, '(A)') BKVAR
        OPEN(5, FILE= BKVAR, STATUS= 'NEW')
      END IF
      DO 10 I=1, IR1
        TK(I)=PTC(I)-PTC(1)
        IF (CT .EQ. 1) THEN
          WRITE(4,*) TK(I), PKKONEONE(I)
        ELSE
          WRITE(5,*) TK(I), PKKONEONE(I)
        END IF
      10 CONTINUE
      RETURN
      END

C
C *****
C      SUBROUTINE WHICH COMPUTES THE PHI MATRIX
C      PHI = I + A*T
C *****
C
      SUBROUTINE PHIDEL(T,N,A,PHI)
      DIMENSION A(6,6), PHI(6,6), COR(6,6), C(6,6)
      F=1.
      DO 100 IR = 1,N
        DO 100 IC = 1,N
          PHI(IR,IC) = 0.
          PHI(IR,IR) = 1.
      100  C(IR,IC) = A(IR,IC)
      110 DO 120 IR = 1,N
        DO 120 IC = 1,N
          COR(IR,IC) = T/F*C(IR,IC)
      120  PHI(IR,IC) = PHI(IR,IC)+COR(IR,IC)
      RETURN
      END

C
C *****
C      THIS SUBROUTINE COMPUTES THE OPTIMUM GAIN MATRIX G(K) AND
C      THE COVARIANCE MATRIX P(K+1/K) BASED ON THE EQUATIONS:
C
C      
$$G(K) = P(K/K-1) * TRANS^0 H(K) * INV^0 [H(K) * P(K/K-1) * TRANS^0 H(K) + R]$$

C
C      
$$P(K/K+1) = PHI * P(K/K) * TRANS^0 PHI + Q(K)$$

C *****
C
      SUBROUTINE GAIN(PKK,PKKM1,Q,R,PHI,H,N,G,HI,ND,MD,LD)
      DIMENSION PKK(6,6), Q(6,6), H(6,6), G(6,6), R(6,6), C(6,6), D(6,6),
      * HI(6,6), HT(6,6), TEMP(6,6), TEMP1(6,6), TEMP2(6,6),
      * PHI(6,6), PHIT(6,6), PKKM1(6,6), TEMP3(6,6), TEMP4(6,6)
      CALL TRANS(H,N,N,HT,ND,MD)
      CALL PROD(PKKM1,HT,N,N,N,TEMP,ND,MD,LD)
      CALL PROD(H,TEMP,N,N,N,TEMP1,ND,MD,LD)

```

```

DO 140 I=1,N
DO 140 J=1,N
140  TEMP3(I,J)=TEMP1(I,J)
    TEMP3(1,1)=TEMP1(1,1)+R(1,1)
    TEMP3(2,2)=TEMP1(2,2)+R(2,2)
    TEMP3(3,3)=TEMP1(3,3)+R(3,3)
    TEMP3(4,4)=TEMP1(4,4)+R(4,4)
    TEMP3(5,5)=TEMP1(5,5)+R(5,5)
    TEMP3(6,6)=TEMP1(6,6)+R(6,6)
DO 143 I=1,N
DO 143 J=1,N
143  C(I,J)=TEMP3(I,J)
    CALL RECIP(C,D,N)
170  CALL PROD(TEMP,D,N,N,N,G,ND,MD,LD)
    CALL PROD(G,H,N,N,N,TEMP,ND,MD,LD)
DO 180 I = 1,6
DO 180 J = 1,6
180  TEMP(I,J) = -TEMP(I,J)
    CALL ADD(HI,TEMP,N,N,TEMP,ND,MD)
    CALL PROD(TEMP,PKKM1,N,N,N,PKK,ND,MD,LD)
    CALL TRANS(PHI,N,N,PHIT,ND,MD)
    CALL PROD(PKK,PHIT,N,N,N,TEMP,ND,MD,LD)
    CALL PROD(PHI,TEMP,N,N,N,TEMP1,ND,MD,LD)
    CALL ADD(TEMP1,Q,N,N,PKKM1,ND,MD)
    RETURN
    END
C
C *****
C  SUBROUTINE WHICH ADDS TWO INPUT MATRICES
C *****
C
C  SUBROUTINE ADD(A,B,N,M,C,ND,MD)
    DIMENSION A(ND,MD),B(ND,MD),C(ND,MD)
    DO 100 I = 1,N
      DO 100 J = 1,M
100  C(I,J) = A(I,J) + B(I,J)
    RETURN
    END
C
C *****
C  SUBROUTINE WHICH SUBTRACTS THE SECOND INPUT MATRIX FROM THE FIRST
C *****
C
C  SUBROUTINE SUB(A,B,N,M,C,ND,MD)
    DIMENSION A(ND,MD),B(ND,MD),C(ND,MD)
    DO 100 I = 1,N
      DO 100 J = 1,M
100  C(I,J) = A(I,J) - B(I,J)
    RETURN
    END
C
C *****
C  SUBROUTINE WHICH MULTIPLIES TWO INPUT MATRICES
C *****
C
C  SUBROUTINE PROD(A,B,N,M,L,C,ND,MD,LD)

```

```

        DIMENSION A(ND,MD),B(MD,LD),C(ND,LD)
        DO 100 I = 1,N
          DO 100 J = 1,L
100      C(I,J) = 0.
          DO 110 I = 1,N
            DO 110 J = 1,L
              DO 110 K = 1,M
110      C(I,J) = C(I,J) + A(I,K)*B(K,J)
        RETURN
        END

C
C *****
C SUBROUTINE WHICH COMPUTES THE TRANSPOSE OF THE INPUT MATRIX
C *****
C
      SUBROUTINE TRANS(A,N,M,C,ND,MD)
        DIMENSION A(ND,MD),C(MD,ND)
        DO 100 I = 1,N
          DO 100 J = 1,M
100      C(J,I) = A(I,J)
        RETURN
        END

C
C *****
C SUBROUTINE WHICH CALCULATES THE RECIPROCAL OF THE INPUT MATRIX
C *****
C
      SUBROUTINE RECIP(A,C,N)
        DIMENSION A(N,N),C(N,N),D(20,20)
        DO 100 I = 1,N
          DO 100 J = 1,N
100      D(I,J) = A(I,J)
          DO 115 I = 1,N
            DO 115 J = N+1,2*N
115      D(I,J) = 0.0
            DO 120 I = 1,N
              J = I + N
120      D(I,J) = 1.0
              DO 240 K = 1,N
                M = K + 1
                IF (K .EQ. N) GOTO 180
                L = K
                DO 140 I = M,N
140      IF (ABS(D(I,K)) .GT. ABS(D(L,K))) L = I
                IF (L .EQ. K) GOTO 180
                DO 160 J = K,2*N
                  TEMP = D(K,J)
                  D(K,J) = D(L,J)
160      D(L,J) = TEMP
                DO 185 J = M,2*N
185      D(K,J) = D(K,J)/D(K,K)
                IF (K .EQ. 1) GOTO 220
                M1 = K - 1
                DO 200 I = 1,M1
                  DO 200 J = M,2*N
200      D(I,J) = D(I,J) - D(I,K)*D(K,J)

```

```

      IF (K .EQ. N) GOTO 260
220 DO 240 I =M,N
      DO 240 J = M,2*N
240 D(I,J) = D(I,J) - D(I,K)*D(K,J)
260 DO 265 I = 1,N
      DO 265 J = 1,N
          K = J + N
265 C(I,J) = D(I,K)
      RETURN
      END
C
C *****
C SUBROUTINE WHICH INITIALIZES SEVERAL MATRICES
C *****
C
      SUBROUTINE INITS
      COMMON W(6,6),XKK(6),PHI(6,6),Q(6,6),XKKM1(6),
      * A(6,6), B(6,6), H(6,6), HI(6,6), PKKM1(6,6),PKK(6,6)
      DO 190 I = 1, 6
          XKK(I) = 0.
          DO 190 J = 1, 6
              Q(I,J) = 0.
              PHI(I,J) = 0.
              A(I,J) = 0.
              B(I,J) = 0.
              H(I,J) = 0.
              HI(I,J) = 0.
              HI(I,I) = 1.
              PKK(I,J) = 0.
              PKK(I,I) = 1.
              PKKM1(I,J)=0.0
              PKKM1(I,I) = 1000000.0
190 CONTINUE
          A(1,2) = 1.
          A(3,4) = 1.
          A(5,6) = 1.
          H(1,1) = 1.
          H(2,3) = 1.
          H(3,5) = 1.
      RETURN
      END
C
C *****
C SUBROUTINE WHICH ALLOWS CHANGING OF W AND PKKM1 TO ALTER THE
C BEHAVIOR OF THE FILTER WITHOUT HAVING TO RE-COMPILE THE PROGRAM
C *****
C
      SUBROUTINE CHANGE(W,PKKM1)
      CHARACTER*1 ASK
      DIMENSION W(6,6),PKKM1(6,6)
      REAL*4 X
      5 FORMAT(A1)
      WRITE(*,10)W(1,1)
10  FORMAT(' W(1,1) IS THIS VALUE: ',F11.5)
      WRITE(*,20)
20  FORMAT(' ENTER (Y) TO CHANGE IT. ')

```

```

READ(*,5)ASK
X=W(1,1)
IF (ASK .EQ. 'Y' .OR. ASK .EQ. 'y') THEN
    WRITE(*,40)
    READ(*,50)X
ENDIF
DO 25 I=1,6
    W(I,I)=X
25 CONTINUE
WRITE(*,30)PKKM1(1,1)
30 FORMAT(' PKKM1(I,I) IS THIS VALUE:',F10.4)
WRITE(*,20)
READ(*,5)ASK
X=PKKM1(1,1)
IF (ASK .EQ. 'Y' .OR. ASK .EQ. 'y') THEN
    WRITE(*,40)
    READ(*,50)X
ENDIF
DO 35 I=1,6
    PKKM1(I,I)=X
35 CONTINUE
40 FORMAT(' ENTER A VALUE FOR ELEMENTS(I,I) WITH A DECIMAL POINT. ')
50 FORMAT(G10.5)
RETURN
END

```

APPENDIX B. INPUT DATA FILE FORMATTING PROGRAM

This program is designed to set up properly formatted files for use as input data by the main program shown in Appendix A. It is an abbreviated version of a program presented in [Ref. 18] which was developed to create usable data files.

The program first reads in the names of the desired input and output files and the array numbers of interest. An option is given the user to remove unwanted headers from the input data file, which some of the available files have, then the target of interest is requested. Finally, the program writes to the output file only those data points that meet the specified criteria or issues an error message if appropriate.

```
C *****
C PROGRAM TO READ IN RAW DATA FROM KEYPORT HYDROPHONE ARRAYS,
C SEGREGATE IT BY USER SPECIFIED MODE, AND RETAIN DATA ONLY
C FROM THE HYDROPHONES SELECTED. INPUT AND OUTPUT FILE NAMES
C ARE PROVIDED BY THE USER.
C *****
C
C CHARACTER DO*2, DSNAME*13, OUTFIL*13
C CHARACTER*3 ENDCHK
C CHARACTER*1 C
C INTEGER PC, ARRAY, NHEAD,A1,A2,NUM
C
C WRITE(*,*) ' ENTER THE NAME OF YOUR INPUT FILE: '
C READ(*,'(A)') DSNAME
C WRITE(*,*) 'INPUT NAME OF DESIRED OUTPUT FILE'
C READ(*,'(A)') OUTFIL
C WRITE(*,*) 'ENTER NUMBERS OF ARRAYS TO BE PAIRED'
C READ(*,*) A1,A2
C OPEN(1,FILE=DSNAME,STATUS='OLD')
C OPEN(2,FILE=OUTFIL,STATUS='NEW')
C
C WRITE(*,*) ' DO YOU WANT TO REMOVE HEADERS? (Y or N)'
C READ(*,'(A)') C
C IF(C.EQ. 'N') GOTO 12
C
11 READ(1,120)ENDCHK
C WRITE(*,*) ENDCHK
C IF(ENDCHK.NE. 'END') GOTO 11
12 CONTINUE
C
C WRITE(*,*) ' INPUT MODE TO BE KEPT?'
C READ(*,*) NUM
C
10 READ(1,100,END=50)PC,DO,X,Y,Z,ARRAY,MODE
```

```

      IF(DO .EQ. 'EV')GOTO 20
      IF(MODE .NE. NUM) GOTO 20
      IF((ARRAY .NE. A1) .AND. (ARRAY .NE. A2)) GO TO 20
20    WRITE(2,110)PC,X,Y,Z,ARRAY,MODE
      CONTINUE
      GOTO 10

```

```

50    WRITE(*,*)' THERE WAS AN ERROR IN THE FILE'
      CONTINUE
100   FORMAT(I5,A2,1X,F7.1,2X,F7.1,2X,F7.1,30X,I2,1X,I2)
110   FORMAT(1X,I5,2X,3F10.1,2X,I2,2X,I2)
120   FORMAT(31X,A3)
      CLOSE (UNIT=1)
      CLOSE(UNIT=2)
      END

```

LIST OF REFERENCES

1. Read, Robert R., "Remote Monitoring Of The Calibration Of A System Of Tracking Arrays," *IEEE Journal Of Oceanic Engineering*, Vol. OE-12, No.1, pp. 198-206, January 1987.
2. Naval Ordnance Systems Command Navord OD 41964, *NAVTORPSTA Keyport Range Complex And Associated Data*, Appendix G, 1 May 1970.
3. Naval Undersea Warfare Engineering Station, *Tracking Technology For Undersea Weapons Ranges*, Presentation IEG-2 to NATO, 5 October 1981.
4. Mitschang, G. W., *An Application Of Nonlinear Theory To Passive Target Location And Tracking*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 1974.
5. Benson, E. J., *An Application Of Kalman Filtering To Underwater Tracking*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1976.
6. Dwyer, D. M., *Real Time Kalman Filtering For Torpedo Range Tracking*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1978.
7. O'Brien, P. A., *An Application Of Kalman Filtering To Torpedo Tracking*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1980.
8. Isik, M., *An Application Of Kalman Filtering And Smoothing To Torpedo Tracking*, Master's Thesis, Naval Postgraduate School, Monterey, California, October 1982.
9. Karaman, S., *Fixed Point Smoothing Algorithm To The Torpedo Tracking Problem*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1986.
10. Gelb, Arthur (ed.), *Applied Optimal Estimation*, M. I. T. Press, 1974.

11. Brown, Robert Grover, *Introduction to Random Signal Analysis And Kalman Filtering*, John Wiley & Sons, Inc., 1983.
12. Maybeck, Peter S., *Stochastic Models, Estimation, And Control Volume 1*, Academic Press, 1979.
13. Meditch, J. S., *Stochastic Optimal Linear Estimation And Control*, McGraw-Hill, Inc., 1969.
14. Maybeck, Peter S., *Stochastic Models, Estimation, And Control Volume 2*, Academic Press, 1982.
15. Moyer, Cleve, Little, John, and Bangert, Steve, *PC-Matlab For MS-DOS Personal Computers, Version 3.2-PC*, MathWorks, Inc., June 8, 1987.
16. Microsoft, *Microsoft Fortran Optimizing Compiler For The MS-DOS Operating System User's Guide*, Microsoft Corporation, 1987.
17. McCracken, Daniel D., *Fortran With Engineering Applications*, John Wiley & Sons, Inc., 1967.
18. Korlu, S., *Monitoring The Calibration Of A Torpedo Test Range*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor H. A. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	5
5.	Professor R. R. Read, Code 55Re Department of Operational Research Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Assistant Professor J. Burl, Code 62B1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Commanding Officer Naval Undersea Warfare Engineering Station Keyport, WA 98345	2
8.	LT. R. B. Nicklas 82-6 Buddington Rd. Groton, CT 06340	2